รายงานฉบับสมบูรณ์

การพัฒนาระบบแสดงผลการใช้พลังไฟฟ้าผ่านอินเตอร์เน็ต
Development of Power Monitoring via Internet System

นายสันติ สถิตวรรธนะ Santi Sathiwantanah
นายพิทักษ์ สถิตวรรธนะ Pitak Sathiwantanah

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย

ได้รับการสนับสนุนทุนวิจัยจากมหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย
งบประมาณเงินรายได้ปีการศึกษา 2559

การพัฒนาระบบแสดงผลการใช้พลังไฟฟ้าผ่านอินเตอร์เน็ต

นายสันติ สถิตวรรธนะ[1] นายพิทักษ์ สถิตวรรธนะ[2]

บทคัดย่อ

งานวิจัยธ์ฉบับนี้จัดทำเพื่อการพัฒนาระบบแสดงผลการใช้พลังงานไฟฟ้าผ่านอินเทอร์เน็ตเพื่อนำไปใช้ในการตรวจวัดการใช้พลังงานไฟฟ้าภายในอาคารโดยระบบสามารถวัดกระแสไฟฟ้าทั้ง 1 เฟส และ 3 เฟส หลักการทำงานของระบบเริ่มต้นจากการวัดกระแสไฟฟ้าผ่านทางอุปกรณ์การวัดรู้ Current transformer จากนั้นนำกระแสที่ได้เข้าวงจรชดเชยแรงดันต่อไปก็นำสัญญาณเข้าสู่อุปกรณ์ไมโครคอนโทรลเลอร์ Arduino UNO R3 เพื่อแปลงค่ากระแสให้เป็นค่าพลังงานไฟฟ้าที่ตัวตรวจรู้ได้ทำการวัดได้หลังจากนั้นทำการส่งข้อมูลผ่านทางระบบเครือข่ายไร้สาย WiFi ไปยังเครื่องแม่ข่ายที่ทำหน้าที่เป็น AP(Access Point) และทำหน้าที่เว็บเซอร์เวอร์ เมื่อเครื่องแม่ข่ายได้รับค่าจากอุปกรณ์ตรวจวัดผ่านทางการส่งข้อมูลแบบ http GET ระบบจะทำการนั้นนำค่าที่ได้จัดเก็บในฐานข้อมูล MySQL โดยจะนำค่าที่ได้รับมาแสดงผลในรูปแบบกราฟผ่านทางระบบอินเตอร์เน็ต โดยผู้ใช้สามารถเข้ามาตรวจสอบผลของการใช้กระแสไฟฟ้าจากระบบผ่านทางเว็บไซด์ได้ สำหรับโปรแกรมในการประมวลผลของระบบแสดงผลได้ใช้โปรแกรมภาษา php ผลการทดสอบผลการวัดการใช้พลังงานกับเครื่องมือการวัดการใช้พลังงานผลปรากฏว่ามีค่าใกล้เคียงกันคิดเป็น 98 เปอร์เซ็น

คำสำคัญ : กระแสไฟฟ้า ตัวตรวจรู้

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลศรีวิชัย อ.เมือง จ.สงขลา 90000

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

พลังงานเป็นปัจจัยที่สำคัญในการตอบสนองความต้องการของประชาชน ภาคธุรกิจและอุตสาหกรรม แต่ประเทศไทยมิได้มีแหล่งพลังงานเชิงพาณิชย์ภายในประเทศมากพอกับความต้องการ ทำให้ต้องพึ่งพาพลังงานจากต่างประเทศเป็นส่วนใหญ่ที่ปัจจุบันมีมูลค่ากว่า 5 แสนล้านบาท แนวทางสำคัญที่จะช่วยลดอัตราการเพิ่มความต้องการใช้พลังงานของประเทศคือการส่งเสริมให้มีการใช้พลังงานอย่างมีประสิทธิภาพและประหยัดในทุกภาคส่วน คณะรัฐมนตรีในการประชุมเมื่อวันที่ 20 มีนาคม 2555 ได้มีมติให้หน่วยงานราชการดำเนินมาตรการลดใช้พลังงานลงให้ได้อย่างน้อย 10% เพื่อเป็นตัวอย่างให้กับภาคเอกชน ภาคประชาชนในการใช้พลังงานอย่างมีประสิทธิภาพ เพื่อผ่อนภาระรายจ่ายด้านการนำเข้าน้ำมันจากต่างประเทศ ถ้าทุกหน่วยงานสามารถลดการใช้พลังงานลงได้ตามเป้าหมายคาดว่าจะลดปริมาณการใช้พลังงานของ 9,290 หน่วยงาน คิดเป็นมูลค่า 950 ล้านบาท (ข้อมูลจาก www.e-report.energy.go.th ณ วันที่ 30 พฤศจิกายน 2558)

ในปัจจุบันได้มีการจัดตั้งคณะทำงานประหยัดพลังงานของหน่วยงานเพื่อร่วมกันพิจารณาปริมาณการใช้ไฟฟ้าภายในหน่วยงาน และร่วมกันพิจารณาวิธีการใช้งานของแต่ละระบบที่ควรจะเหมาะสมตามความจำเป็นและพิจารณาวิธีการดูแลบำรุงรักษาที่ควรจะมีการมอบหมายบุคคลและกำหนดเวลาในการตรวจสอบอย่างน้อยปีละ 2 ครั้ง เพื่อนำมาจัดทำแผนและมาตรการประหยัดพลังงานที่เหมาะสมกับหน่วยงาน โดยกำหนดเป้าหมายลดการใช้พลังงานลงให้ได้อย่างน้อยร้อยละ 10 เมื่อเทียบกับค่าการใช้พลังงานมาตรฐาน แต่ยังขาดอุปกรณ์ที่ใช้ในการตรวจสอบพลังงานไฟฟ้า ซึ่งทางผู้จัดทำโครงการได้มองเห็นถึงปัญหาในการตรวจสอบค่าพลังงานไฟฟ้า จึงได้จัดทำ “อุปกรณ์วัดการใช้กำลังงานไฟฟ้าแบบไร้สาย” จะทำให้อุปกรณ์อิเล็กทรอนิกส์สามารถเชื่อมต่อกันได้ เช่น โทรศัพท์มือถือ พีดีเอ คอมพิวเตอร์ส่วนบุคคล ผู้จัดทำโครงการมองเห็นถึงความสามารถของเทคโนโลยีดังกล่าวที่จะนำมาใช้ในการทำโครงการเพื่อเป็นเครื่องมือในการวัดผล โดยสามารถทำการตรวจสอบผ่านอุปกรณ์สมาร์ทโฟน แท็บเล็ต หรืออุปกรณ์ต่างๆ ที่มีความสามารถในการเชื่อมต่อระบบอินเตอร์เน็ตหรือไวไฟ

ทางผู้วิจัยได้พัฒนาเพราะเห็นถึงปัญหาในจุดนี้ จึงได้จัดทำ “อุปกรณ์วัดการใช้พลังงานไฟฟ้าผ่านระบบอินเตอร์เน็ต” เพื่อเป็นการนำความรู้ที่ได้เรียนมาผสมผสานเข้ากับเทคโนโลยีที่ใช้ ให้เกิดประโยชน์ และเพื่อช่วยวัดปริมาณการใช้พลังงานไฟฟ้าภายในอาคาร อีกทั้งยังใช้ต้นทุนในการผลิตที่ต่ำมากเมื่อเทียบกับอุปกรณ์ตรวจวัดกระแสไฟฟ้าที่นำเข้ามาจากต่างประเทศ ภายในอาคาร

1.2 วัตถุประสงค์

1.2.1 เพื่อออกแบบระบบการแสดงผลการใช้พลังงานไฟฟ้าในมหาวิทยาลัย

1.2.2 เพื่อพัฒนาอุปกรณ์ต้นแบบในการวัดและแสดงผลการใช้พลังงานไฟฟ้า

1.2.3 เพื่อพัฒนาระบบการแสดงผลการใช้พลังงานไฟฟ้าผ่านระบบอินเทอร์เน็ต

1.3 ขอบเขต

1.3.1 อุปกรณ์ต้นแบบสำหรับวัดการใช้พลังงานไฟฟ้าที่ติดตั้งในอาคาร

1.3.2 ระบบการแสดงผลการใช้พลังงานไฟฟ้าผ่านทางอินเตอร์เน็ต

1.3.3 ลการใช้พลังงานจากอาคารที่ได้ทำการทดสอบระบบ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 นำไปถ่ายทอดเทคโนโลยีให้กับคณะและหน่วยงานที่สนใจในการวัดค่าการใช้พลังงานเพื่อนโยบายประหยัดพลังงาน

1.4.2 จัดทำคู่มือและการนำไปใช้ประโยชน์ให้กับหน่วยงานภายนอกที่สนใจ

บทที่ 2
งานวิจัยและทฤษฎีที่เกี่ยวข้อง

2.1 อุปกรณ์ตรวจรู้กระแสไฟฟ้า Current Transformer

หม้อแปลงเครื่องมือวัด (Instrument Transformer) เป็นเครื่องสำเร็จ (Apparatus) ที่ใช้สำหรับแปลงแรงดันหรือกระแส เพื่อให้สามารถใช้ได้กับเครื่องวัดไฟฟ้าภายในพิสัยความถี่กำลัง โดยเครื่องวัดปกติ การขยายพิสัยวัดกระแสทำได้โดยใช้ชันต์การขยายพิสัยวัดแรงดันทำโดยใช้ตัวต้านทานอนุกรม (ตัวคูณ) ซึ่งสามารถทำได้โดยสะดวก และพบเห็นทั่วไปในการวัดกระแสตรง อย่างไรก็ตาม วิธีนี้มีขีดจำกัดที่ค่ากระแสและแรงดันไม่สูงมาก เมื่อกระแสมีค่าสูงมากเกินไม่กี่ร้อยแอมแปร์ กำลังที่สูญเสียในชันต์จะมีค่ามากพอ นอกจากนั้นจะต้องคิดถึงการฉนวนเครื่องวัดให้พอเพียง ซึ่งเป็นเรื่องยากถ้ากระแสที่จะวัดอยู่ที่หลายร้อยหรือหลายพันโวลต์เหนือดิน ในกรณีที่ต้องวัดค่าที่มีพิสัยกว้างให้มีความถูกต้อง เป็นเรื่องที่ไม่เหมาะสมที่จะต้องมีเครื่องวัดที่สามารถวัดค่าพิสัยกว้างให้ถูกต้องทั้งหมด แต่โดยการลดกระแสหรือแรงดันค่าสูงลงมาด้วยอัตราส่วนที่รู้ค่า และถูกต้องหลายๆค่าโดยใช้หม้อแปลง และใช้เครื่องมือวัดที่สามารถวัดขนาดกระแสหรือแรงดันค่าไม่สูง แต่มีความถูกต้องมากวัดค่า จะสามารถได้พิสัยกว้าง โดยมีความถูกต้องตามต้องการแต่ประหยัดและปลอดภัย

เราสามารถแบ่งหม้อแปลงเครื่องมือวัดตามการใช้งานได้เป็นหม้อแปลงกระแส (Current Transformer (CT)) และหม้อแปลงแรงดัน (Potential Transformer (PT)) หม้อแปลงเครื่องมือวัดมีลักษณะเหมือนหม้อแปลงกำลังทั่วๆไป (สามารถใช้วงจรสมมูลของหม้อแปลงธรรมดาได้) แต่มีข้อพิเศษที่แตกต่าง คือ

1. จำเป็นต้องรู้อัตราส่วนจำนวนรอบอย่างแน่นอน เนื่องจากค่าความผิดพลาดหนึ่งรอบใน 200 ถ้าพิจารณาในลักษณะหม้อแปลงกำลัง จะทำให้เกิดความแตกต่างของแรงดันน้อยมาก แต่ก็จะผิดพลาดมาก ถ้าพิจารณาในการนำไปใช้ด้านการวัด

2. แรงดันตกคร่อมในขดลวดจะต้องมีค่าต่ำที่สุด เพื่อหลีกเลี่ยงการเลื่อนเฟสหรือการเปลี่ยนอัตราส่วนการแปลง โดยการออกแบบหม้อแปลงให้มีค่ารีแอกแตนซ์ต่ำ และโดยใช้สายตัวนำ (ทองแดง) ให้มีขนาดโตกว่าที่ต้องการ ดังนั้นในกรณีหม้อแปลงกำลัง การโหลดจะถูกจำกัดโดยผลเนื่องจากความร้อน แต่การโหลดของหม้อแปลงเครื่องมือวัดจะถูกจำกัดโดยค่าความถูกต้อง

เราอาจแบ่งหม้อแปลงเครื่องมือวัดตามความถูกต้องออกได้เป็น 5 ชั้น คือ 0.1 0.2 0.5 1.0 และ 3.0 สามชั้นแรกใช้สำหรับการทดสอบหรืองานวัดละเอียด สองชั้นหลังจะใช้สำหรับการวัดทั่วไป

หม้อแปลงกระแส (Current Transformer (CT)) ในแอมมิเตอร์กระแสตรงแบบ Permanent Magnet Moving Coil (PMMC) เมื่อต้องการขยายพิสัยวัดจะทำโดยการใช้ชันต์เพื่อแบ่งกระแสที่ต้องการวัดระหว่างเครื่องวัดกับชันต์ วิธีนี้จะเหมาะสำหรับวงจรกระแสตรง แม้ว่าเมื่อกระแสเพิ่มมากขึ้น กำลังสูญเสียในชันต์จะมีค่ามาก ในวงจรกระแสสลับ การแบ่งกระแสจะไม่เพียงขึ้นกับความ

ต้านทานของเครื่องวัดกับชันต์เท่านั้น แต่ยังขึ้นกับค่ารีแอกแตนซ์ของมันด้วย เพราะว่าการวัดกระแสสลับอาจกระทำในพิสยความถี่กว้าง ดังนั้นจะเป็นการยากที่จะได้ความถูกต้องสูง หม้อแปลงกระแสจะทำให้เกิดการขยายพิสัยที่ต้องการผ่านอัตราส่วนการแปลง และทำให้เกิดค่าที่อ่านเกือบจะเหมือนกัน โดยไม่คำนึงถึงค่าคงที่(ความต้านทาน หรือรีแอกแตนซ์) หรือจำนวนของเครื่องวัด (ภายในขอบเขต) ที่ต่ออยู่ในวงจร

หลักการเบื้องต้นของ CT

กระแสโหลดที่ต้องการวัด จะไหลผ่านขอปฐมภูมิ ซึ่งอาจจะเป็นลวดตัวนำเส้นเดียว ถือว่าเป็นหนึ่งรอบทางปฐมภูมิ ขดลวดทุติยภูมิจะมีจำนวนรอบมากกว่า และจะต่อกับมาตรวัดกระแส ขดลวดของรีเลย์หรือขดลวดของวัตต์มิเตอร์ การทำงานของหม้อแปลงกระแสจะขึ้นอยู่กับสมดุลของค่าแอมแปร์ - รอบที่สร้างขึ้น โดยขดปฐมภูมิและทุติยภูมิ ถ้าหม้อแปลงเป็นอุดมคติ คือ ไม่มีกระแสทำแม่เหล็ก (Magnetizing Current) หรือความสูญเสียในแกน จะได้

$$\frac{I_p}{I_s} = n_{ct}$$

n_{ct} = คือ อัตราส่วนรอบของหม้อแปลงกระแสโดย

ปกติจะออกแบบให้ขดลวดทุติยภูมิจ่ายกระแสทุติยภูมิขนาด 5 A. แผ่นป้ายชื่อบนตัวจะกำหนดอัตราส่วนของหม้อแปลง เช่น 500:5 A. ค่านี้ไม่ใช่อัตราส่วนรอบ เพียงแต่แสดงว่า กระแสปฐมภูมิ 500 A. จะให้กระแสทุติยภูมิ 5 A. เพื่อต่อขดทุติยภูมิเข้ากับแอมมิเตอร์พิสัย 5 A. เพราะว่าโหลดในระบบจะกำหนดกระแสปฐมภูมิ กระแสทุติยภูมิจะสัมพันธ์กับกระแสปฐมภูมิ โดยอัตราส่วนรอบผกผัน (โดยประมาณ) ในการสร้าง จะต้องทำให้กระแสทำแม่เหล็ก ความสูญเสียในแกน และเส้นแรงรั่วซึมน้อยที่สุด เพื่อจะแน่ใจว่า อัตราส่วนกระแสปฐมภูมิต่อทุติยภูมิจริงๆ จะเข้าใกล้อัตราส่วนรอบผกผัน จะมีความผิดพลาดที่สำคัญในหม้อแปลงกระแสคือ ความผิดพลาดของกระแส (หรืออัตราส่วน) กับความผิดพลาดมุมเฟส มีการนิยามความผิดพลาดกระแสหรืออัตราส่วน (Current or Ratio Error) ว่าเท่ากับ

$$\frac{(I_p/I_s)_1 - (I_p/I_s)_2}{(I_p/I_s)_2} \times 100\%$$

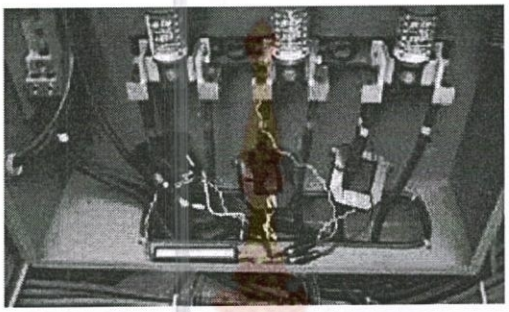
$(I_p/I_s)1$ คือ อัตราส่วนตามพิกัด

$(I_p/I_s)2$ คือ อัตราส่วนจริง

ความผิดพลาดมุมเฟส คือ มุมเฟสระหว่างเวกเตอร์ของกระแสปฐมภูมิกับเวกเตอร์ของกระแสทุติยภูมิที่กลับทิศ สำหรับหม้อแปลงที่สมบูรณ์ ค่ามุมเฟสนี้จะเป็น 0 ค่าความผิดพลาดเหล่านี้ จะกำหนดโดยเทียบกับโหลดที่ต่ออยู่กับขดทุติยภูมิโดยเฉพาะค่าหนึ่ง เรียกโหลดนี้ว่า เบอร์เดน

(Burden) ของหม้อแปลง และกำหนดเบอร์เดนนี้โดยกำลังปรากฏ (VA) ที่ใช้ไปในโหลดภายใต้ความถี่และกระแสทุติยภูมิที่กำหนด และโดยตัวประกอบกำลัง (Power Factor) ของมัน

ข้อควรระวังในการใช้หม้อแปลงกระแส



รูปที่ 2.1 ตัวอย่างการใช้เครื่องมือวัดกระแส

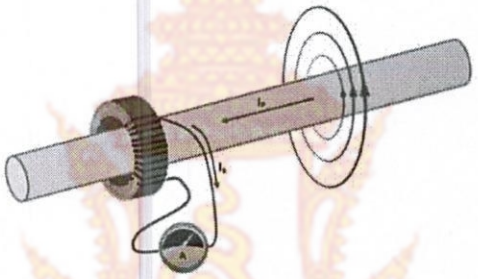
จะต้องไม่เปิดวงจรทุติยภูมิของหม้อแปลงกระแสขณะที่มีกระแสไหลทางด้านปฐมภูมิ จำนวนแอมแปร์-รอบจะถูกทำให้คงที่โดยกระแสปฐมภูมิ และจะไม่ลดลงเมื่อทางทุติยภูมิเปิดวงจร การเปิดวงจรทางด้านทุติยภูมิจะลดแอมแปร์-รอบทางด้านทุติยภูมิเป็นศูนย์ ซึ่งจะไม่มีแรงเคลื่อนแม่เหล็กกลับไปต่อต้านแรงเคลื่อนแม่เหล็กจากแอมแปร์-รอบปฐมภูมิ และความหนาแน่นของเส้นแรงจะเพิ่มขึ้นจนกระทั่งแกนอิ่มตัว ผมที่ตามมาของการที่เส้นแรงอิ่มตัว จะกระทำต่อจำนวนรอบทางทุติยภูมิที่มีค่ามาก ทำให้แรงดันที่ถูกเหนี่ยวนำในขดลวดทุติยภูมิจะมีค่าสูง ซึ่งอาจเป็นอันตรายต่อผู้เปิดวงจรหรือทำความเสียหายต่อฉนวนของหม้อแปลง นอกจากนั้น จะเกิดความร้อนเนื่องจากความสูญเสียในแกนขณะที่มันอิ่มตัว อาจมากพอที่จะทำลายหม้อแปลงได้ หม้อแปลงกระแสที่ใช้งานจะมีหลายลักษณะ เช่น

- เครื่องวัดแบบ Clip-on เป็น CT แบบหนึ่ง โดยแกนเหล็กสามารถแยกออกจากกัน เพื่อให้สอดเข้าไปวัดสายตัวนำ (ทำให้สามารถวัดค่ากระแสโดยไม่จำเป็นต้องตัดวงจร)

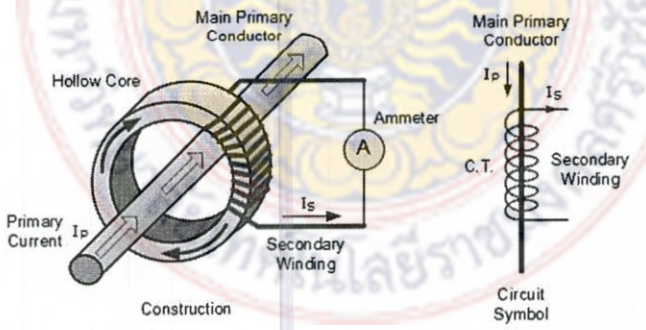
-โพรบกระแส (Current Probe) ของออสซิลโลสโคปสำหรับใช้วัดกระแสสลับ จะทำงานในลักษณะเดียวกัน โดยมีแกนแยกจากกันที่ปลายของโพรบ

CT ที่ใช้ในห้องปฏิบัติการส่วนมาก จะมีลักษณะดังรูป คือจะมีรูตรงกลางให้สายตัวนำพันรอบ โดยทางปฐมภูมิจะมีหลายจุดแยก ทางด้านทุติยภูมิจะมีจำนวนรอบคงที่

รูปที่ 2.2 แสดง Current Transformer แบบต่างๆ



รูปที่ 2.3 แสดงหลักการวัดกระแสโดยใช้ Current transformer



รูปที่ 2.4 แสดงสัญลักษณ์ของ Current Transformer

Current Transformer เป็นการวัดกระแสไฟฟ้าทางอ้อมเช่นกัน โดยใช้หลักการเหนี่ยวนำของสนามแม่เหล็กเหมือนกับหม้อแปลงไฟฟ้า แต่เปลี่ยนให้ฝั่ง Primary เป็นสายไฟที่ต้องการวัดกระแสแทน และมีเพียงขดลวดฝั่ง Secondary เรียกว่า Current Transformer เมื่อเราจ่ายกระแสไฟฟ้าสลับไหลผ่านสายไฟ จะทำให้เกิดเส้นสนามแม่เหล็กเปลี่ยนแปลงไปมา และไปตัดกับขดลวดที่พันรอบ

แกน Inductive Sensor ทำให้เกิดกระแสไฟฟ้าขึ้นเมื่อต่อกับโหลด ซึ่งจะวัดได้เฉพาะกระแสไฟฟ้า AC เท่านั้น กรณีที่จ่ายกระแสไฟฟ้า DC เข้าไปในสายไฟ จะไม่มีการเปลี่ยนแปลงของสนามแม่เหล็ก ซึ่งจะไม่เกิดการเหนี่ยวนำของสนามแม่เหล็ก หลักการนี้สามารถนำไปใช้กับ Clampmeter

รูปที่ 2.5 แสดง Current Transformer แบบต่างๆ

รูปที่ 2.6 แสดง Clamp Meter

ในกรณีที่เราวัดกระแสไฟฟ้าที่ฝั่ง Secondary ได้ต่ำมาก เป็นเพราะโหลดกินกระแสน้อยมากเมื่อเทียบกับย่านที่ Current Transformer วัดได้ เช่น นำ Current Transformer ที่เหมาะสำหรับวัดในย่านกระแส 100 A ไปใช้วัดกระแสที่หลอดไฟใช้ ซึ่งมีปริมาณน้อยมาก เมื่อเทียบกับย่านที่เซ็นเซอร์วัดได้ ผู้ใช้ต้องเพิ่มสัญญาณให้มากกว่านี้ วิธีแก้ไขหนึ่งคือการ เพิ่มรอบขดลวดในฝั่ง Primary เพื่อลดอัตรา Turn Ratio ลง จากสูตร

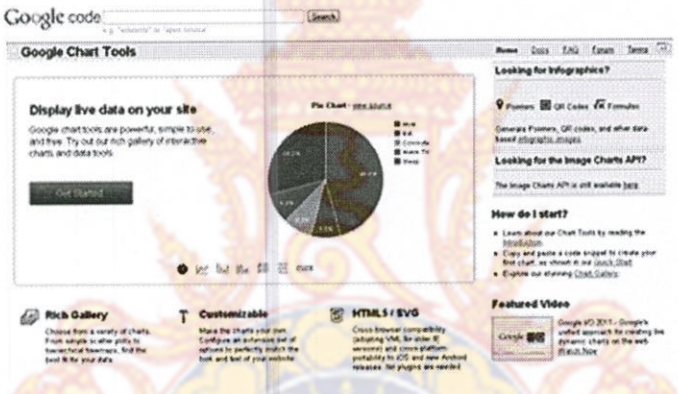
$$T \cdot R = \frac{N_p}{N_s} = \frac{I_p}{I_s}$$

โดย N_p = รอบขดลวดในฝั่ง Primary

N_s = รอบขดลวดในฝั่ง Secondary
I_p = กระแสผ่านขดลวดฝั่ง Primary
I_s = กระแสผ่านขดลวดฝั่ง Secondary

2.2 ความรู้เรื่อง google charts API

Google Charts เป็นเครื่องมือที่ในการสร้างแผนภูมิรูปภาพ หรือที่เราเรียกว่ากราฟ (Graphs) หรือชาร์ต (Charts) ที่เราเอาไว้นำเสนอรายงานต่างๆ โดยบริการของ Google Charts นี้สามารถเรียกใช้ในรูปแบบของ Visualization API หรือส่วนต่อประสานโปรแกรมของ Google ที่จะแปลงข้อมูลจากฐานข้อมูลสถิติต่างๆ จากเว็บไซต์ของคุณให้แสดงผลออกมาเป็นรูปแบบแผนภูมิที่ เรียบง่ายไปจนถึงรูปแบบที่มีลำดับขั้นของข้อมูลที่ซับซ้อน หรือมีขนาดใหญ่ โดยอาศัยหลักการเชื่อมต่อสื่อสารส่ง-รับ ข้อมูลบนสถาปัตยกรรมอินเทอร์เน็ตแบบ Client-Server



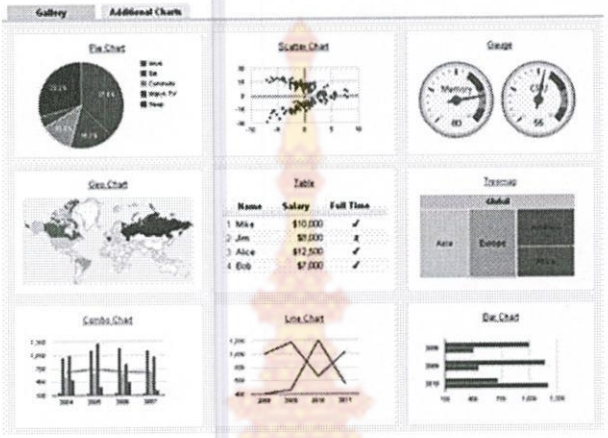
รูปที่ 2.7 แสดง google charts API

ภายใต้ชุดพัฒนาของส่วนต่อประสานโปรแกรมอย่าง Visualization API ของ Google ทำให้สามารถนำชุดข้อมูลสถิติที่อยู่ในรูปของตาราง (Spread Sheets) มาทดสอบผ่านหน้าจอทดสอบก่อนนำไปใช้งานร่วมกับบริการ Google Docs หรือประยุกต์ให้ซับซ้อนขึ้นมาหน่อยโดยการพัฒนาร่วมกับการเขียนโปรแกรมเพื่อดึงข้อมูลจากฐานข้อมูลขึ้นมาเป็นค่าตัวแปร เพื่อให้แสดงผลเป็นกราฟ หรือแผนภูมิที่สวยงามผ่านโปรแกรมท่องเว็บไซต์อย่างเว็บบราวเซอร์ได้ทุกประเภทที่เป็นเว็บบราวเซอร์มาตรฐาน

รูปแบบของบริการ Google Charts

รูปแบบแผนภูมิรูปภาพ หรือกราฟ ที่สร้างขึ้นจาก API ของ Google Charts นั้นพัฒนาจากภาษา JavaScript ทำให้สะดวกในการนำไปใช้กับเอกสารประเภท HTML หรือเอกสารสำหรับแสดงหน้าเว็บไซต์ทุกประเภท อีกทั้ง Google Charts ยังมีรูปแบบของกราฟ หลากหลายรูปแบบให้เลือกนำไปใช้ในการประกอบหน้าจอรายงานสถิติให้เหมาะสมกับชุดข้อมูล และสถานการณ์ที่ต้องนำข้อมูล

ไปใช้ เช่น Pie chart , Line Charts , Bars Charts และรูปแบบต่างๆ อีกมากมายทำให้ง่ายต่อกลุ่มผู้พัฒนาเว็บไซต์ที่สามารถนำ Scripts ในการแสดงผลของกราฟที่ Google Charts สร้างขึ้นมานำไปใช้ร่วมกับรูปแบบดีไซน์ภายในเว็บไซต์ของตน



รูปที่ 2.8 แสดง google charts API รูปแบบต่าง ๆ

รูปแบบกราฟ หรือแผนภูมิของ Google Charts สามารถเลือกปรับรูปแบบให้มีการโต้ตอบ หรือเอนิเมชันเล็กน้อย พร้อมกับระบบ Dashboard ที่ง่ายต่อการบริหารจัดการกราฟ และเก็บชุดข้อมูลกราฟที่คุณได้สร้างไว้บนบริการ Google Charts เพื่อความสะดวกในการกลับมานำ Script ของ Google Charts ไปใช้บนหน้าเว็บไซต์ของคุณได้ตลอดเวลา นอกจากรูปแบบการโต้ตอบ และ Dashboard ในการควบคุมแล้ว มาตรฐานของภาษาโปรแกรมบน Script ของ Google Charts ที่สร้างขึ้นนั้นยังรองรับเทคโนโลยีเปิดในอนาคตอย่าง HTML5 และยังทำงานข้ามแพลตฟอร์มไปแสดงผลบนหน้าจอสมาร์ทโฟนระบบปฏิบัติการ Andriod , ระบบปฏิบัติการ iOS ของ Apple อย่าง iPhone และ iPad โดยไม่ต้องติดตั้งส่วนเสริม หรือ Plug-in เพิ่มลงในสมาร์ทโฟนเลย

2.3 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์ไอซี (Integrated Circuit, IC) ที่สามารถโปรแกรมการทำงานได้ซับซ้อน สามารถรับข้อมูลในรูปสัญญาณดิจิทัลเข้าไปทำการประมวลผลแล้วส่งผลลัพธ์ข้อมูลดิจิทัลออกมาเพื่อนำไปใช้งานตามที่ต้องการได้

2.3.1 ไมโครคอนโทรลเลอร์ Arduino รุ่น UNO R3

เป็นไมโครคอนโทลเลอร์ตระกูล AVR โดยใช้ไมโครคอนโทลเลอร์เบอร์ATmega 328 ขนาด 8 บิต เป็น MCU ประจำบอร์ดโดย MCU รุ่นนี้มีขา pin ทั้งหมด 28 ขาและมีจุดเด่นคือเป็นไมโครคอนโทรลเลอร์ขนาดเล็กแต่เพียบพร้อมไปด้วยทรัพยากรพื้นฐานต่างๆ อย่างครบถ้วน จึงมีความเหมาะสมเป็นอย่างยิ่งในการใช้งานทั่วไปสำหรับภายในมีทั้งระบบฮาร์ดแวร์ของ SPI, UART, I2C, Watchdog,Timer/Counter ,PWM เป็นต้น และสามารถใช้ในการพัฒนาโปรแกรมด้วย Arduino ได้ทันที

รูปที่ 2.9 ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino UNO R3

คุณสมบัติของบอร์ดไมโครคอนโทรลเลอร์

ชุดโมดูลบอร์ดไมโครคอนโทรลเลอร์ Arduino มี ATmega328 ขนาด 8 บิต เป็น MCU ตระกูล AVR ประจำบอร์ด โดยเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาแบบ Crystal Oscillator ที่มีค่า 16 MHz เพื่อให้สามารถใช้งานพอร์ทสื่อสารอนุกรมได้อย่างลงตัว

1) บอร์ดสามารถเปลี่ยนการติดตั้ง MCU เป็นแบบ 28 pin หรือเบอร์อื่นในอนุกรมเดียวกันได้โดยไม่ต้องมีการดัดแปลงใดๆ เช่น ATmega88 เป็นต้น

2) มีหน่วยความจำแบบ Flash 32 Kbyte โดยแบ่งเป็น Boot loader 2 Kbyte

3) มีEPROM 1Kbyte/SRAM 2 Kbyte

4) มีพอร์ตเป็น 14 digital input/output pins ซึ่งมี 6 pin สามารถสร้างเป็น PWM outputs

5) มีพอร์ต 6 Analog input/output pins

6) ไฟกระแสตรง (DC) ขา I/O pin มีค่า 40 mA

7) ไฟกระแสตรง (DC) ขา 3.3 V pin มีค่า 50 mA

8) MCU ประจำบอร์ดที่ได้รับการติดตั้ง Boot loader สามารถอัพโหลดโค้ดให้บอร์ดผ่านทางพอร์ตสื่อสารแบบอนุกรมได้ทันที

9) มีขั้วต่อ USB Interface สื่อสารข้อมูลแบบอนุกรมเข้าคอมพิวเตอร์ได้ทันที

10) มีLED สำหรับแสดงสถานะไฟเลี้ยง

11) มีLED แสดงสถานการณ์รับส่งข้อมูล

12) ใช้ไฟเลี้ยงประจำบอร์ด 7-12 Volt

จุดเด่นของ Arduino

"Arduino" เป็นภาษาอิตาลี ซึ่งเป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open source ที่ได้รับการปรับปรุงมาจากโครงการ Open source ของ AVR Arduino มี

การใช้งานด้วยรูปแบบที่ง่ายไม่ซับซ้อน ซึ่งแม้ว่า Arduino จะมีรูปแบบการใช้งานคล้ายกับไมโครคอนโทรลเลอร์อย่าง Basic Stap ของ Parallax แต่ก็มีจุดเด่นกว่ารายอื่นคือ

1) ราคาไม่แพง เนื่องจากมี Source Code และวงจรแจกให้ฟรีสามารถต่อวงจรขึ้นมาใช้เองได้รวมถึงมีการเปิดเผยวงจร Source code ทั้งหมดทำให้สามารถนำไปพัฒนาต่อยอดได้ดีทั้งด้านฮาร์ดแวร์และซอฟแวร์

2) โปรแกรมที่ใช้พัฒนาของ Arduino สามารถรองรับการทำงานทั้ง Window, Linux และ OSx

3) รูปแบบคำสั่งง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริงๆ กับส่วนที่มีความซับซ้อนมากและยังสามารถสร้างคำสั่งรวมถึง Library ใหม่ๆ ขึ้นมาใช้งานเองได้

เปรียบเทียบภาษาซีกับ Arduino

สำหรับการเขียนโปรแกรมของ Arduio นั้นใช้ภาษา C++ ซึ่งเป็นรูปแบบของภาษาซีประยุกต์รูปแบบหนึ่งที่มีโครงสร้างการทำงานของตัวภาษาโปรแกรมโดยรวมคล้ายกับ ภาษาซีมาตรฐานทั่วไป เพียงแต่ได้มีการปรับปรุงเพื่อลดความยุ่งยากในการใช้งานลดลงและให้ผู้ใช้สามารถใช้งานเขียนโปรแกรมได้ง่าย สะดวกมากกว่าการเขียนภาษาซีแบบมาตรฐาน แต่ในความเป็นจริงนั้นโปรแกรมดังกล่าวไม่ใช่ C-compiler โดยตรงเนื่องจาก Arduino จะมีลักษณะการทำงานเช่นเดียวกับ Text Editor ของภาษา C++ ตัวหนึ่งโดยจะทำงานร่วมกับ Utility บางส่วนที่ Arduino สร้างขึ้นมารองรับโดย Arduino จะใช้รูปแบบ การทำงานของEditor เป็นฉากหน้าในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริงๆ แล้ว Arduino จะไปเรียกใช้ตัวแปลภาษาซีและ Utility อื่นที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ตระกูล AVR อีกทีหนึ่ง Arduino จะเลือกใช้คอมไพลเลอร์ของ “GNU AVR-GCC Toolchain” ร่วมกับ Library functionของ “avr-libc” ส่วน Utility ที่ใช้ในการอัพโหลดโค้ดให้กับ AVR จะใช้ในส่วนของ “AVRDude”

โครงสร้างการเขียนโปรแกรมภาษาซีของ Arduino

ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมเป็นส่วนย่อยๆ หลายๆ ส่วนโดยเรียกแต่ละส่วนว่า “ฟังก์ชั่น” และเมื่อนำฟังก์ชันมารวมเข้าด้วยกันก็จะเรียกว่า “โปรแกรม” โดยโครงสร้างการเขียนโปรแกรมของ Arduino ทุกๆ โปรแกรมจะประกอบไปด้วยฟังก์ชั่นจำนวนเท่าใดก็ได้แต่อย่างน้อยสุดต้องมี 2 ฟังก์ชั่นคือSetup () และLoop ()

1) Setup (): เป็นฟังก์ชั่นบังคับที่ต้องกำหนดให้มีทุกๆ โปรแกรม ถึงแม้ว่าบางโปรแกรมจะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้เสมอเพียงแต่ไม่ต้องเขียนคำสั่งใดๆ ไว้หลังวงเล็บปีกกา {} ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชั่น โดยฟังก์ชันนี้จะใช้สำหรับบรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียว ตอนเริ่มต้นทำงานของโปรแกรมครั้งแรกเท่านั้น ซึ่งได้แก่คำสั่งเกี่ยวกับการ Setup ค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่ของการใช้งานของ PinMode และค่า Baudrate สำหรับการใช้งานสื่อสารพอร์ตอนุกรม เป็นต้น

2) Loop (): เป็นส่วนฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุกโปรแกรมเช่นเดียวกันฟังก์ชั่น Setup () โดยฟังก์ชั่น Loop () นี้จะใช้ในการบรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวนรอบซ้ำๆ กันไปไม่รู้จบ ซึ่งเปรียบเทียบได้กับฟังก์ชั่น main () ใน ANSCI-C นั่นเอง

ทฤษฎีไมโครคอนโทรลเลอร์ตระกูล AVR รุ่น ATmega 328 ซึ่งเป็นสถาปัตยกรรมชั้นสูงแบบ RISC (Reduce Instruction Set Computer)

1) RISC คือตัวทำให้การประมวลผลมีความเร็ว 1 คำสั่ง/1 clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS/MHz

2) ชุดคำสั่ง 131 คำสั่งต่อ1 รอบนาฬิกา

3) รีจิสเตอร์ขนาด 8 บิต 32 ตัว

4) ความเร็วในการประมวลผลมากกว่า 20 ล้านคำสั่งต่อวินาที(MIPS) ที่ 20 MHz

โดยมีคุณสมบัติต่างๆ ดังต่อไปนี้

หน่วยความจำ

1) แบบ 32 Kbyte สามารถเขียนลบโปรแกรมได้ 10,000 ครั้ง

2) แบบ EPROM 1 Kbyte สามารถเขียนและลบโปรแกรมได้ 100,000 ครั้ง

3) แบบ SRAM 2 Kbyte

ไฟเลี้ยง

1) ระหว่าง1.8 ถึง2.5 v

ความถี่สัญญาณนาฬิกา

2) ระหว่าง 0 ถึง 4 MHz

3) มีการรองรับอุปกรณ์ต่อพ่วง

4) อุปกรณ์สื่อสารข้อมูลแบบอนุกรมแบบ I2C และ USART

อื่นๆ

1) มีระบบ Reset แบบอัตโนมัติเมื่อจ่ายกระแสไฟฟ้าเข้าไมโครคอนโทรลเลอร์

2) มีฟังก์ชั่นตรวจสอบแรงดัน

3) มีระบบการขัดจังหวะทั้งภายในและภายนอก

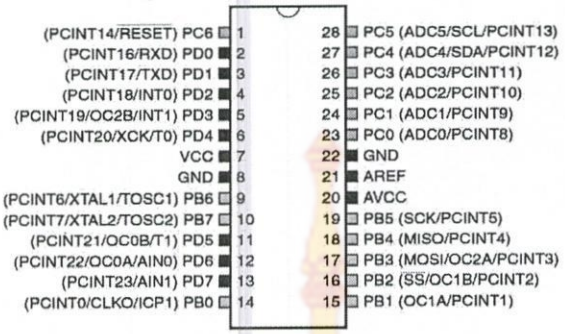
4) มีระบบตรวจจับความผิดพลาดของ CPU

5) มีโหมดอนุรักษ์พลังงาน 5 mode ได้แก่ Idle, ADC Noise Reduction, Power-Save,Power-Down, Standby

6) มีการรองรับอุปกรณ์ต่อพ่วง

7) อุปกรณ์สื่อสารข้อมูลแบบอนุกรมแบบ I2C และ USART

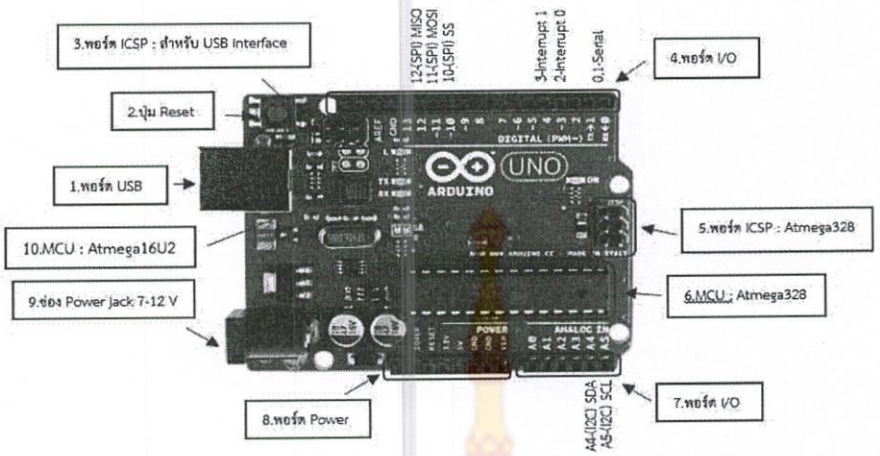
ส่วนประกอบต่างๆ ของไอซีATmega 328



รูปที่ 2.10 ไอซี ATmega 328

ตารางที่ 2.1 ฟังก์ชั่นการทำงานของ ATmega 328

ชื่อ	รายละเอียด	ขา
GND	ขากราวด์ต่อสายดิน	8,22
VCC	ไฟเลี้ยง 1.8 ถึง 5.5v	7
Port B (PB 7: 0) XTAL1/XTAL2/TOSC1/ TOSC2	1. เป็นพอร์ต 2 ทิศทางขนาด 8 บิต โดยสามารถ กำหนดให้ขาของแต่ละพอร์ตสามารถตั้งค่าให้ Pull up Resistor ได้(ภายในเป็นอิสระแยกจากกัน เพื่อดึงแรงดันของลอจิก1 ให้เท่ากับ 5 v) 2. สามารถใช้งานพิเศษตามความต้องการของ ATmega 328 โดยขึ้นอยู่กับการตั้งค่าสัญญาณ นาฬิกาที่ขา PB6 ที่ใช้เป็นแรงดัน Oscillator และขาอินพุตของวงจรสัญญาณ Clock Oscillator	9,10, 14-19
PC6/RESET	ขา Reset	1
Port D (PD7:0)	1. เป็นพอร์ตสองทิศทางขนาด 8 บิต โดยสามารถ กำหนดให้ขาของแต่ละพอร์ตสามารถตั้งค่าให้ Pull up Resistor ได้ 2. สามารถใช้งานพิเศษตามความต้องการของ ATmega 328	1-6, 11-13
AVCC	ใช้จ่ายไฟให้กับวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล มักจะต่อเข้ากับขาVCC	20
AREF	แรงดันอ้างอิงที่ใช้งานในส่วนของวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลมักต่อกับ VCC	21
ADC7:6 (TQFP and QFN/MLF Package Only)	ขากำลังงานใช้แปลงสัญญาณอนาล็อกเป็นดิจิตอล	23-28
Port C (PC5:0)	1. เป็นพอร์ตสองทิศทางขนาด 8 บิต โดยสามารถกำหนดให้ขาของแต่ละพอร์ตสามารถตั้งค่าให้ Pull up Resistor ได้ 2. สามารถใช้งานพิเศษตามความต้องการของ ATmega 328	23-28



รูปที่ 2.11 แสดงรูปแบบและพอร์ตต่างๆ ของ Arduino UNO R3

2.3.2 ทฤษฎีเกี่ยวกับโหมดประหยัดพลังงาน (Power and Sleep Mode)

ไมโครคอนโทรลเลอร์ในปัจจุบันมีโหมดการทำงานหลายโหมด เช่น โหมดการทำงานปรกติ (Normal mode) และมีโหมดอื่นๆ เช่น โหมดการประหยัดพลังงาน (Power Saving Mode หรือ Low-Power Mode) ซึ่งจะใช้พลังงานต่ำกว่าโหมดปรกติ และอาจมีโหมดการประหยัดพลังงานให้เลือกใช้ได้หลายระดับ การประหยัดพลังงานทำได้หลายวิธี เช่น การหยุดการทำงานของ CPU ภายใน การหยุดการทำงานของส่วนอื่นที่ไม่ใช่ CPU อย่างเช่น UART, ADC เป็นต้น หรือใช้เทคนิคอื่นอย่างเช่น การปรับความเร็วของสัญญาณ Clock หรือปรับระดับแรงดันไฟเลี้ยงให้ลดต่ำลง

โหมดประหยัดพลังงานเหล่านี้มีความสำคัญ โดยเฉพาะอย่างยิ่งในกรณีที่ต้องการนำไมโครคอนโทรลเลอร์ไปประยุกต์ใช้งานและจำเป็นต้องใช้พลังงานจากแบตเตอรี่ การใช้กำลังไฟฟ้าหรือพลังงานที่ลดลง จะช่วยเพิ่มอายุการใช้งานแบตเตอรี่ (Battery Life Time Extension) ได้นานขึ้น หรือในบางกรณี ไมโครคอนโทรลเลอร์ไม่จำเป็นต้องทำงานตลอดเวลา อาจจะทำงานบางอย่างตามช่วงเวลาที่กำหนดไว้ เป็นระยะๆ ดังนั้นในช่วงเวลาที่ไม่ต้องทำอะไร จึงสามารถเข้าสู่โหมดการประหยัดพลังงาน เช่น โหมดที่ทำให้ MCU "หลับ" หรือที่เรียกว่า Sleep Mode

เมื่อไมโครคอนโทรลเลอร์อยู่ในโหมด Sleep วิธีการปลุกให้ตื่นขึ้นมาทำงานต่อจากเดิมที่หยุดค้างไว้ มีอยู่หลายวิธี ทั้งนี้ก็ขึ้นอยู่กับตระกูลของไมโครคอนโทรลเลอร์ที่เลือกใช้งาน เช่น ปลุกโดยการสร้างอินเตอร์รัพต์จาก Watchdog Timer (WDT) ปลุกโดยการสร้างอินเตอร์รัพต์จากภายนอก (External Interrupt) เป็นต้น

	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X[2]	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X[2]	X[3]	X	X[2]	X	X	X		
Power-down								X[3]	X				X		X
Power-save					X		X[2]	X[3]	X	X			X		X
Standby[1]						X		X[3]	X				X		X
Extended Standby					X[2]	X	X[2]	X[3]	X	X			X		X

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT1 and INT0, only level interrupt.

รูปที่ 2.12 ตารางแสดงการปลุกไมโครคอนโทรลเลอร์ที่อยู่ในโหมด Sleep

การใช้กำลังไฟฟ้าของไมโครคอนโทรลเลอร์ในโหมด Sleep จะต่ำกว่าในโหมด Normal โดยทั่วไปการวัดระดับการใช้กำลังไฟฟ้าของวงจรอิเล็กทรอนิกส์ ที่ใช้แรงดันไฟเลี้ยงกระแสตรงคงที่ (regulated DC voltage supply) สามารถทำได้โดยการวัดปริมาณกระแสที่จ่ายให้วงจรหรือโหลดดังกล่าว (Current Consumption) และกำลังไฟฟ้าที่ใช้ไปในขณะนั้น จะแปรผันตามปริมาณกระแสที่วัดได้ ดังนั้นปริมาณกระแสจึงถูกนำมาใช้เป็นตัวบ่งชี้ระดับการใช้พลังงานหรือกำลังไฟฟ้าในแต่ละช่วงของวงจรอิเล็กทรอนิกส์ ซึ่งก็รวมถึงบอร์ดไมโครคอนโทรลเลอร์ อย่างเช่น Arduino

Arduino ที่ใช้ชิป Atmega328 มีโหมดการประหยัดพลังงานอยู่ 6 โหมด ซึ่งเรียงลำดับจากโหมดที่รักษาพลังงานได้น้อยสุดไปยังโหมดรักษาพลังการได้มากสุด ดังนี้

1) SLEEP_MODE_IDLE: ปริมาณกระแสที่ใช้งานในขณะอยู่ในโหมดนี้คือ 15 mA
2) SLEEP_MODE_ADC: ปริมาณกระแสที่ใช้งานในขณะอยู่ในโหมดนี้คือ 6.5 mA
3) SLEEP_MODE_PWR_SAVE: ปริมาณกระแสที่ใช้งานในขณะอยู่ในโหมดนี้คือ 1.62 mA
4) SLEEP_MODE_EXT_STANDBY: ปริมาณกระแสที่ใช้งานในขณะอยู่ในโหมดนี้คือ 1.62 mA
5) SLEEP_MODE_STANDBY: ปริมาณกระแสที่ใช้งานในขณะอยู่ในโหมดนี้คือ 0.84 mA
6) SLEEP_MODE_PWR_DOWN: ปริมาณกระแสที่ใช้งานในขณะอยู่ในโหมดนี้คือ 0.36 mA

ตัวอย่างโค้ดโปรแกรม

```
void enterSleep(void){
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable();
  power_all_disable();
  sleep_mode();
  sleep_disable();
  power_all_enable();
}
```

คำอธิบายโปรแกรม

set_sleep_mode(SLEEP_MODE_PWR_DOWN);

เป็นฟังก์ชั่นในการเลือกโหมดทำงานของ sleep mode ในที่นี้ได้เลือกการทำงาน sleep mode แบบ SLEEP_MODE_PWR_DOWN เพราะเป็นโหมดที่รักษาพลังงานได้ดีที่สุด

sleep_enable();

เป็นฟังก์ชั่นเปิดการทำงาน sleep mode

power_all_disable();

เป็นฟังก์ชั่นหยุดใช้พลังงานของโมดูลทั้งหมด

sleep_mode();

เป็นฟังก์ชั่นให้ sleep mode ทำงานเพื่อการประหยัดพลังงาน

sleep_disable();

เป็นฟังก์ชั่นปิดการทำงาน sleep mode

power_all_enable();

เป็นฟังก์ชั่นเปิดใช้พลังงานของโมดูลทั้งหมด

ทฤษฎีสัญญาณเฝ้าระวัง (Watchdog Timer)

คุณสมบัติ

1) มีสัญญาณนาฬิกาแยกอิสระจากสัญญาณออสซิเลเตอร์ของระบบ ทำให้ทำงานอย่างต่อเนื่องแม้อยู่ใน sleep mode

2) มีการโหมดการทำงาน 3 แบบ คือ

2.1) อินเตอร์รับ

2.2) รีเซตระบบ

2.3) อินเตอร์รับและรีเซตระบบ

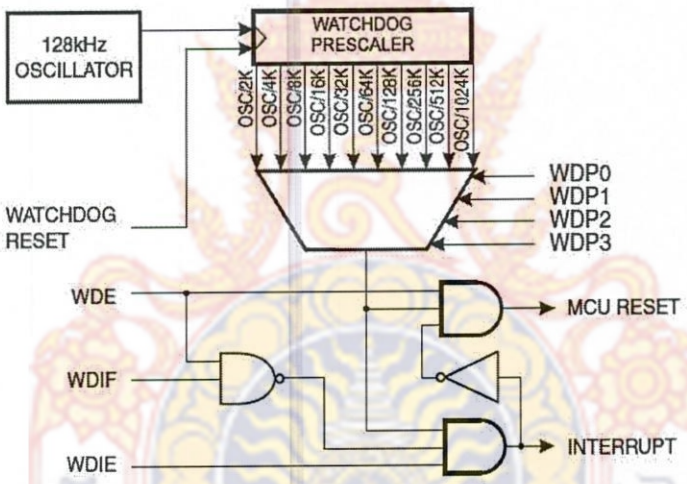
3) สามารถเลือกช่วงเวลาในการทำงานได้ตั้งแต่ 16 mS ไปจนถึง 8 S

4) ถ้าเป็นไปได้ฮาร์ดแวร์ควรมีการกำหนด Configuration bit (Fuse) Wactchdog อยู่เสมอ (WDTON) เพื่อป้องกันการทำงานผิดพลาด

ข้อมูลทั่วไป

ATmega48A/PA/88A/PA/168A/PA/328/P ได้มีการปรับปรุงให้มี Watchdog Timer (WDT) ภายใน WDT เป็นตัวตั้งเวลานับรอบสัญญาณนาฬิกาที่แยกออกมาจากชิบโดยมีความถี่ออสซิเลเตอร์อยู่ที่ 128kHZ

WDT จะช่วยในการ Interrupt(สัญญาณขัดจังหวะ) หรือรีเซ็ตการทำงานของระบบเมื่อการนับเวลาถึงค่าเวลาที่กำหนด ในโหมดการทำงานปกติที่จำเป็นต้องมีระบบที่ใช้ WDR นั้น แนะนำให้ใช้ Watchdog Timer Reset(wdt_reset()) ในการเริ่มการทำงานของตัวนับใหม่ก่อนที่ระบบจะรีเซ็ต หากระบบไม่เริ่มตัวนับเวลา Interrupt หรือการรีเซ็ตระบบ นั้นตัวนับเวลาก็จะสิ้นสุดลง



รูปที่ 2.13 โครงสร้าง Watchdog Timer

ในโหมดอินเตอร์รัพต์ WDT จะช่วยให้การขัดจังหวะเมื่อตัวจับเวลาการใช้งานสิ้นสุด อินเตอร์รัพต์นี้สามารถใช้ในการกระตุ้นการทำงานของอุปกรณ์ให้ออกจาก sleep-modes และยังเป็นการตั้งเวลาของระบบทั่วไป ตัวอย่างหนึ่งคือการจำกัดระยะเวลาสูงสุดที่อนุญาตสำหรับการดำเนินการบางอย่างซึ่งช่วยให้ท่านได้มีการเปิดใช้งานอินเตอร์รัพต์รันนานกว่าที่คาดไว้ ในโหมดการรีเซ็ตระบบจะให้รีเซ็ต WDT เมื่อตัวจับเวลาสิ้นสุด โดยทั่วไปจะใช้เพื่อป้องกันไม่ให้ระบบค้าง กรณีที่มีโค๊ดที่ควบคุมไม่ได้ โหมดที่สาม อินเตอร์รัพต์และโหมดการรีเซ็ตระบบ ได้รวมสองโหมดนี้ไว้ด้วยกันเป็นครั้งแรกโดยให้มีการขัดจังหวะแล้วสลับไปยังโหมดการรีเซ็ตระบบ ยกตัวอย่างโหมดนี้จะทำการปิดระบบที่ปลอดภัยโดยการบันทึกพารามิเตอร์ที่สำคัญก่อนที่จะมีการรีเซ็ตระบบ

Watchdog ต้องมีการกำหนด Configuration bit (Fuse) เปิดไว้เสมอ (WDTON) หากมีการตั้งโปรแกรมไว้จะเป็นการบังคับให้ Watchdog Timer ทำงานในโหมดรีเซ็ตระบบ ด้วย fuse โปรแกรม บิตของโหมดรีเซ็ตระบบ (WDE) และบิตของโหมดอินเตอร์รัพต์ (WDIE) จะถูกกำหนดให้มีค่าเป็น 1 และ 0 ตามลำดับ เพื่อเป็นการรักษาความปลอดภัยโปรแกรมให้ปรับเปลี่ยนการตั้งค่า Watchdog จะต้องทำตามลำดับที่ตั้งเวลาไว้ ตามลำดับสำหรับการเคลียร์ WDE และการเปลี่ยนการตั้งค่าเวลาสิ้นสุด การกำหนดค่าเป็นดังนี้

1) ในการทำงานในลักษณะเดียวกับการเขียนลอจิกที่หนึ่งไปยังอีกที่หนึ่งบิตทำการเปลี่ยนแปลง Watchdog (WDCE) และ WDE ได้ ลอจิกหนึ่งจะต้องได้รับการเขียนขึ้นเพื่อ WDE โดยไม่คำนึงถึงค่าก่อนหน้าของบิต WDE

2) ภายในอีก 4 รอบสัญญาณนาฬิกาให้เขียน WDE และบิต Watchdog prescaler (WDP) ตามที่ต้องการแต่พร้อมด้วยการเคลียร์บิต WDCE ซึ่งต้องดำเนินการในการทำงานหนึ่งครั้ง

คำอธิบายรีจิสเตอร์

1) MCUSR – MCU Status Register

สถานะรีจิสเตอร์ของ MCU มีข้อมูลเกี่ยวกับการรีเซต ซึ่งก่อให้เกิดการรีเซต MCU

Bit	7	6	5	4	3	2	1	0	
0x34 (0x54)	–	–	–	–	WDRF	BORF	EXTRF	PORF	MCUSR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

รูปที่ 2.14 โครงสร้างของ MCUSR

Bit 7-4: Reserved

บิตเหล่านี้เป็นบิตที่ไม่ได้ใช้งานใน ATmega48A/PA/88A/PA/168A/PA/328/P และจะมีค่าเป็นศูนย์เสมอ

Bit 3 – WDRF: Watchdog System Reset Flag

บิตนี้จะตั้งค่า หาก Watchdog มีการรีเซตระบบจะเกิดขึ้นได้ บิตจะถูกรีเซ็ตโดยการเปิดเครื่องหรือรีเซ็ตโดยการเขียนลอจิกศูนย์ไปยัง Flag

Bit 2 – BORF: Brown-out Reset Flag

บิตนี้จะตั้งค่าหาก Brown-out Reset เกิดขึ้น บิตจะถูกรีเซ็ตโดยการเปิดเครื่องหรือรีเซ็ตโดยการเขียนลอจิกศูนย์ไปยัง Flag

Bit 1 – EXTRF: External Reset Flag

บิตนี้จะตั้งค่าหาก External Reset เกิดขึ้น บิตจะถูกรีเซ็ตโดยการเปิดเครื่องหรือรีเซ็ตโดยการเขียนลอจิกศูนย์ไปยัง Flag

Bit 0 – PORF: Power-on Reset Flag

บิตนี้จะตั้งค่าหาก Power-on Reset เกิดขึ้น บิตจะถูกรีเซ็ตโดยการเขียนลอจิกศูนย์ไปยัง Flag เท่านั้น

เพื่อทำให้การใช้งานของ Reset Flags เพื่อระบุเงื่อนไขการรีเซ็ต ผู้ใช้ควรโปรแกรมให้อ่านและตั้งค่ารีเซ็ท MCUSR เร็วที่สุด หาก register จะถูกล้างก่อนการรีเซ็ตเกิดขึ้น สามารถตรวจสอบแหล่งที่มาของการรีเซ็ต ได้โดยการตรวจสอบการ Reset Flags

2) WDTCSR – Watchdog Timer Control Register

Bit	7	6	5	4	3	2	1	0	
(0x60)	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	WDTCSR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	X	0	0	0	

รูปที่ 2.15 โครงสร้างของ WDTCSR

Bit 7 – WDIF: Watchdog Interrupt Flag

บิตนี้จะตั้งค่าเมื่อเกิดการสิ้นสุด Watchdog Timer และ Watchdog Timer ถูกกำหนดค่าสำหรับการอินเตอร์รัพต์ WDIF จะถูกเคลียร์โดยฮาร์ดแวร์เมื่อดำเนินการอินเตอร์รัพต์ที่สอดคล้องกันการจัดการเวกเตอร์ อีกวิธีหนึ่งคือ WDIF จะถูกเคลียร์โดยการเขียนลอจิกหนึ่งไปยัง Flag เมื่อมีการตั้งค่า I-bit ใน SREG และ WDIE, Watchdog ก็จะสิ้นสุดการดำเนินการอินเตอร์รัพต์

Bit 6 – WDIE: Watchdog Interrupt Enable

เมื่อบิตนี้ถูกกำหนดขึ้นและค่า I-bit จะถูกตั้งค่าในสถานะรีจิสเตอร์ เพื่อเปิดใช้งานอินเตอร์รัพต์ Watchdog WDE จะถูกเคลียร์ ในกรณีที่ใช้ร่วมกับการตั้งค่านี้ให้ Watchdog Timer ที่อยู่ในโหมดอินเตอร์รัพต์ และ อินเตอร์รัพต์ที่สอดคล้องกันจะทำงานหากมีการสิ้นสุด Watchdog Timer เกิดขึ้น หากมีการตั้งค่า WDE Watchdog Timer จะอยู่ในโหมดอินเตอร์รัพต์และโหมดการรีเซ็ตระบบ เมื่อ Watchdog Timer สิ้นสุดการทำงานในครั้งแรกก็จะส่งค่า WDIF ออกมา การดำเนินเนินการเกี่ยวกับ interrupt vector จะเคลียร์ค่า WDIE และ WDIF โดยอัตโนมัติด้วยฮาร์ดแวร์(Watchdog จะเข้าสู่โหมดรีเซ็ทระบบ) การดำเนินการนี้จะมีประโยชน์ในการรักษาความปลอดภัยของ Watchdog Timer ในขณะที่ใช้งานอินเตอร์รัพต์ เมื่อยังคงอยู่ในโหมดอินเตอร์รัพต์และโหมดการรีเซ็ตระบบ WDIE จะต้องได้รับการตั้งค่าทุกครั้งหลังจากที่ทำการอินเตอร์รัพต์ แต่อย่างไรก็ตามไม่ควรกระทำให้เสร็จภายในการอินเตอร์ตัวเองอยู่เป็นประจำ และการดำเนินการนี้อาจส่งผลกระทบต่อความปลอดภัยของการทำงาน Watchdog ของ โหมดรีเซ็ทระบบ หากไม่ดำเนินการอินเตอร์รัพท์ก่อนที่จะสิ้นสุดการ watchdog ในครั้งถัดไประบบก็จะทำการรีเซ็ทระบบ

WDTON	WDE	WDIE	Mode	Action on Time-out
1	0	0	Stopped	None
1	0	1	Interrupt Mode	Interrupt
1	1	0	System Reset Mode	Reset
1	1	1	Interrupt and System Reset Mode	Interrupt, then go to System Reset Mode
0	x	x	System Reset Mode	Reset

Note: 1. WDTON Fuse set to "0" means programmed and "1" means unprogrammed.

รูปที่ 2.16 การกำหนดค่า Watchdog Timer

Bit 4 – WDCE: Watchdog Change Enable

บิตนี้จะถูกใช้ในลำดับการกำหนดเวลาสำหรับการเปลี่ยนแปลง WDE และบิต prescaler การเคลียร์ บิต WDE และ/หรือ เปลี่ยนบิต prescaler WDCE จะต้องได้รับการตั้งค่าได้เมื่อเขียนหนึ่งครั้งฮาร์ดแวร์จะเคลียร์ค่า WDCE หลังจากผ่านไปสี่รอบสัญญาณนาฬิกา

Bit 3 – WDE: Watchdog System Reset Enable

WDE จะถูกแทนที่โดย WDRF ใน MCUSR ซึ่งหมายความว่า WDE จะมีการตั้งค่าไว้เมื่อตั้งค่า WDRF อยู่เสมอ การเคลียร์ WDE จำเป็นต้องเคลียร์ WDRF ก่อนเสมอ คุณสมบัตินี้ช่วยให้มั่นใจได้ในระหว่างการรีเซ็ตหลายเงื่อนไขที่ทำให้เกิดความผิดพลาด และมีความปลอดภัยในการเริ่มโปรแกรมใหม่หลังจากเกิดความผิดพลาด

Bit 5, 2-0 - WDP [3:0]: Watchdog Timer Prescaler 3, 2, 1 and 0

บิต WDP[3:0] จะกำหนดช่วงเวลา Watchdog Timer เมื่อ Watchdog Timer ทำงานอยู่ ค่า prescaling ต่างๆ และระยะสิ้นสุดในการทำงานของ Watchdog Timer ควรตั้งให้สอดคล้องกัน ดังแสดงในตาราง

WDP3	WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at V_{CC} = 5.0V
0	0	0	0	2K (2048) cycles	16 ms
0	0	0	1	4K (4096) cycles	32 ms
0	0	1	0	8K (8192) cycles	64 ms
0	0	1	1	16K (16384) cycles	0.125 s
0	1	0	0	32K (32768) cycles	0.25 s
0	1	0	1	64K (65536) cycles	0.5 s
0	1	1	0	128K (131072) cycles	1.0 s
0	1	1	1	256K (262144) cycles	2.0 s
1	0	0	0	512K (524288) cycles	4.0 s
1	0	0	1	1024K (1048576) cycles	8.0 s
1	0	1	0	Reserved	
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

รูปที่ 2.17 ตารางช่วงเวลาในการทำงาน Watchdog Timer

ตัวอย่างโค้ดโปรแกรม

```
void WDT_start() {
  cli();
  MCUSR &= ~(1 << WDRF);
  WDTCSR |= (1 << WDIF);
  WDTCSR |= (1 << WDCE) | (1 << WDE);
  WDTCSR = (1 << WDIE) | (1 << WDP3 );
  wdt_reset();
  sei(); // enable Global interrupt
}
```

คำอธิบายโปรแกรม

cli(); cli- Clear Global Interrupt Flag เป็นฟังก์ชั่นในการหยุดการทำงานของอินเตอร์รัพต์

MCUSR &= ~(1 << WDRF); เคลียร์ค่า รีเซ็ต WDT

WDTCSR |= (1 << WDIF); เคลียร์ค่า รีเซ็ตอินเตอร์รัพต์ WDT

WDTCSR |= (1 << WDCE) | (1 << WDE); ตั้งค่าลอจิกหนึ่งให้บิต WDCE และ WDE

WDTCSR = (1 << WDIE) | (1 << WDP3); ตั้งค่าการสิ้นสุดการทำงานของ WDT ไว้ 4 วินาที และเปิดใช้ WDT โหมดอินเตอร์รัพต์

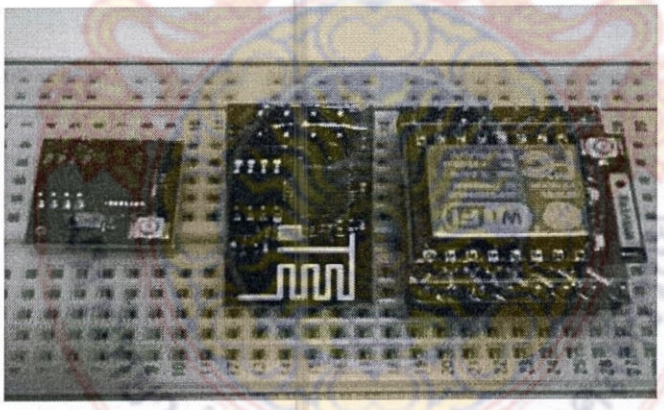
wdt_reset(); กำหนดให้ WatchDog timer เริ่มนับใหม่ (Clear Watchdog)

sei(); SEI - Set Global Interrupt Flag เป็นฟังก์ชั่นในการเปิดการทำงานของอินเตอร์รัพต์

2.4 โมดูลสื่อสารไร้สาย ESP8266

โมดูล ESP8266 เช่น ESP-01 เริ่มออกสู่ตลาดในปี ค.ศ. 2014 และได้รับความนิยมอย่างรวดเร็วในกลุ่มนักพัฒนา เพราะเป็นโมดูลประเภท "Serial-to-Wi-Fi" ที่มีราคาถูกเมื่อเปรียบเทียบกับสินค้าอื่นๆ ในประเภทเดียวกัน เริ่มแรกโมดูลถูกออกแบบมาให้ใช้งานร่วมกับไมโครคอนโทรลเลอร์หรืออุปกรณ์ประเภทอื่น ผ่านขา Tx / Rx แบบ Serial และใช้ชุดคำสั่งแบบ AT Commands เพื่อสื่อสารข้อมูลผ่านเครือข่ายไร้สาย (Wi-Fi) และเชื่อมต่อไปสู่อินเทอร์เน็ตได้

โมดูลนี้ใช้ชิปประเภท SoC (System-on-Chip) ซึ่งมีโปรเซสเซอร์ขนาด 32 บิต "Xtensa LX3" ของบริษัท Tensilica เป็นตัวประมวลผลหลัก ถัดจากโมดูลรุ่น ESP-01 เริ่มมีโมดูลรุ่นอื่นออกมา เช่น ESP-02, ESP-03 ... ESP-13 เป็นต้น ซึ่งมีรูปแบบและขนาดของโมดูลที่แตกต่างกัน จำนวนขา I/O รูปแบบของสายอากาศที่ใช้ เป็นต้น



รูปที่ 2.18 ตัวอย่างโมดูล ESP8266

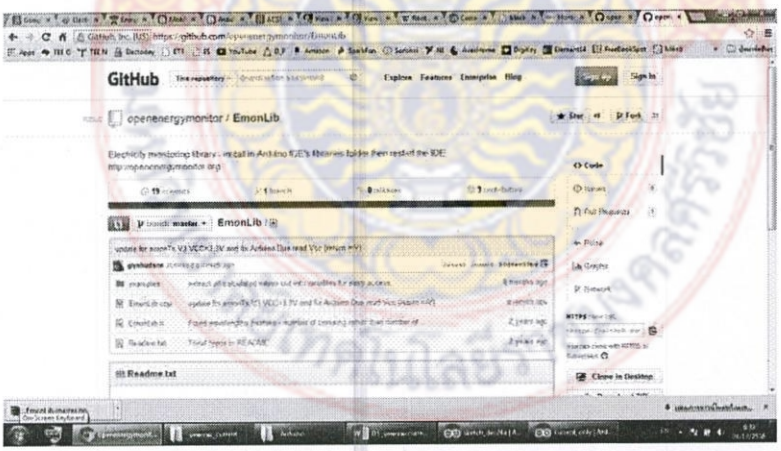
2.5 แนะนำ Emon Library

Emon Library เป็น Library ที่ใช้ในโปรเจค OpenEnergyMonitor โดยใช้ Arduino เป็น MCU อ่านค่าสัญญาณ โดยมีฟังก์ชั่นคำนวณค่า Irms และ Vrms ง่ายต่อการใช้งาน

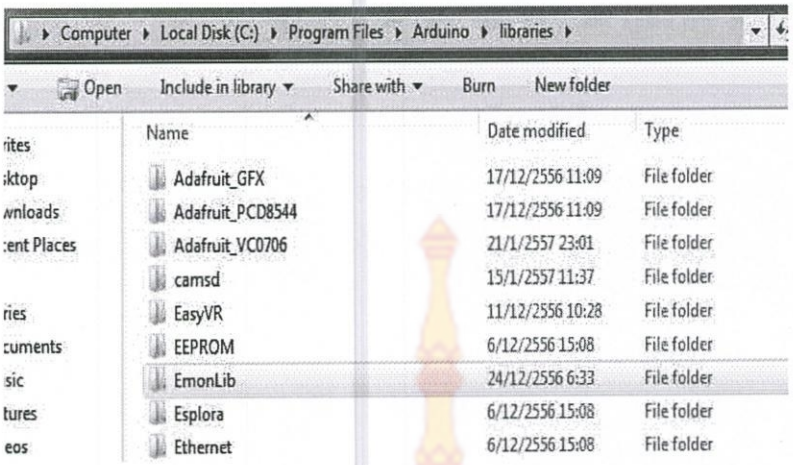
http://openenergymonitor.org/emon/

https://github.com/openenergymonitor/EmonLib

รูปที่ 2.19 หน้าเว็บไซต์โปรเจค OpenEnergyMonitor [3]



รูปที่ 2.20 โหลด Emon Library จากหน้าเว็บไซต์ GitHub [3]



รูปที่ 2.21 ติดตั้ง Library โดยนำโฟลเดอร์ EmonLib

ตัวอย่างโค้ดโปรแกรม

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance
void setup()
{
  Serial.begin(9600);
  emon1.current(1, 11.8);              // Current: input pin, calibration.
}
void loop()
{
  double Irms = (emon1.calcIrms(1480)-0.02);  // Calculate Irms only
   Serial.print(Irms*230.0);         // Apparent power
   Serial.print(" ");
   Serial.println(Irms);             // Irms
}
```

อธิบายโค้ดโปรแกรม

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance
```

เป็นการดึงคำสั่งจาก Library EmonLib มาใช้ โดยใช้คำสั่ง #include "EmonLib.h" จากนั้นประกาศ Instance ชื่อ emon1

```
void setup()
{
  Serial.begin(9600);
```

```
    emon1.current(1, 11.8);            // Current: input pin, calibration.
  }
```

ในฟังก์ชั่น Setup ก่อนที่จะเริ่มอ่านค่าจากเซ็นเซอร์ ให้เปิดใช้งาน Hardware Serial โดยกำหนด Baud Rate เป็น 9600 bps กำหนดค่าให้ฟังก์ชั่น Current ซึ่งประกอบด้วย ขาอนาล็อกที่เท่าไหร่จากโค้ดเป็น 1 คือขา A1 และ 11.8 คือค่าที่ทำการ Calibrate แล้ว

จากโค้ดจะเห็นได้ว่า

```
  void loop()
  {
    double Irms = (emon1.calcIrms(1480)-0.02);  // Calculate Irms only
     Serial.print(Irms*230.0);         // Apparent power
     Serial.print(" ");
     Serial.println(Irms);             // Irms
  }
```

ในฟังก์ชั่น Loop วนอ่านสัญญาณอนาล็อกจากเซ็นเซอร์โดยใช้ฟังก์ชั่น calIrms โดยใส่พารามิเตอร์เป็น จำนวนการอ่าน และเก็บค่าไว้ในตัวแปร Irms จากนั้นส่งค่าตัวแปรแสดงผ่านพอร์ต Serial

บทที่ 3
วิธีการดำเนินงาน

3.1 บทนำ

การออกแบบเป็นปัจจัยหนึ่งที่จะต้องทำในการประดิษฐ์หรือสร้างอุปกรณ์วัดการใช้กำลังงานไฟฟ้าแบบไร้สาย ซึ่งเป็นกระบวนการหนึ่งที่เป็นรูปแบบของการทำงานที่เป็นระบบโดยอาศัยความคิดสร้างสรรค์ที่ได้จากการนำปัญหามาทำการคิดวิเคราะห์แล้วแก้ไขปัญหานั้นทำให้มองเห็นภาพของสิ่งที่จะทำซึ่งมีลักษณะรูปแบบที่แน่นอน การออกแบบเป็นขั้นตอนอันดับแรกก่อนการสร้างตัวชิ้นงานขึ้นมาทำให้มีเป้าหมายที่แน่นอนว่าจะสร้างอะไร เพื่อที่จะนำมาประกอบขึ้นเป็น ตัวชิ้นงานทำให้ไม่เสียเวลาเพราะบางครั้งถ้าไม่มีการออกแบบสิ่งที่สร้างขึ้นมาอาจจะทำงานไม่ได้ตามเป้าหมายที่วางไว้

ในการออกแบบอุปกรณ์วัดการใช้กำลังงานไฟฟ้าแบบไร้สายจำเป็นต้องคำนึงถึงลักษณะของโครงสร้างที่มีอยู่เดิม และรูปแบบการใช้งานทั้งของที่มีอยู่เดิมและที่ติดตั้งเพิ่มทำให้ลงตัวมากขึ้น ทำให้มีทันสมัยมากขึ้น ซึ่งมีขั้นตอนในการออกแบบและกระบวนการทำงานต่างๆ ดังนี้

3.1.1 ศึกษาและรวบรวมข้อมูลเกี่ยวกับ Sensor วัดกระแสไฟฟ้า

3.1.2 ศึกษาและรวบรวมข้อมูลเกี่ยวกับ ESP-02

3.1.3 การออกแบบโครงสร้างของกล่องอเนกประสงค์สำหรับใส่แผงวงจรและอุปกรณ์ต่างๆ ภายใน

3.1.4 การออกแบบโครงสร้างและการจัดทำแผงวงจร

3.2 แผนการดำเนินงาน

ในการดำเนินงานได้ตั้งเป้าหมายของการดำเนินงาน กำหนดขั้นตอนของวิธีการออกแบบระบบขั้นตอนการพัฒนาและติดตั้งอุปกรณ์ และการรอของอุปกรณ์ที่สั่งซื้อ อุปกรณ์บางชนิดที่ไม่มีสินค้าในสต็อกสินค้าทำให้ต้องรออุปกรณ์ในการสั่งซื้อ จึงต้องใช้เวลานานและไม่สามารถปฏิบัติงานต่อเนื่องได้เริ่มดำเนินการเดือนพฤศจิกายน พ.ศ. 2559 ถึง เดือนสิงหาคม พ.ศ. 2560

ตาราง 3.1 แผนการดำเนินงานของโครงงาน

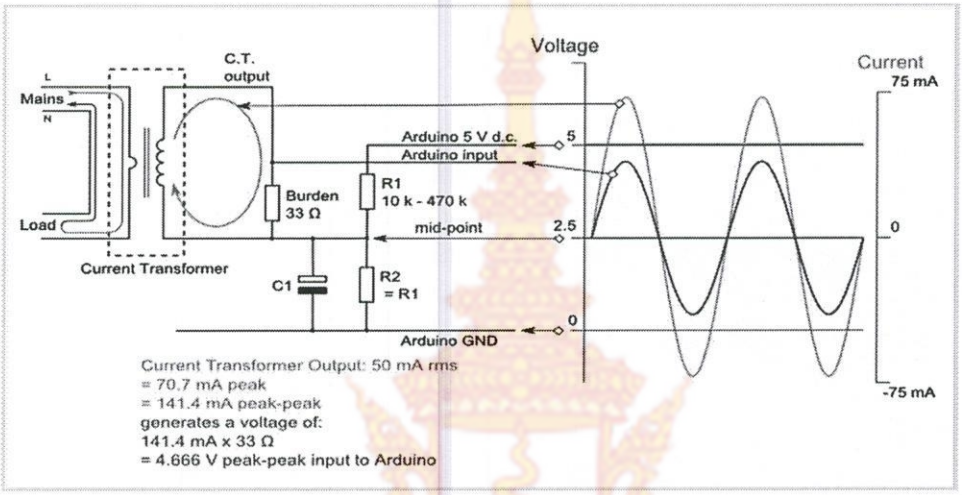
กิจกรรม		ต.ค.-พ.ย 2558	ธ.ค. 2558	ม.ค.-ธ.ค 2559	ม.ค. 2559	ก.พ.-มิ.ย 2560	ก.ค. 2560	ส.ค. 2560
1. นำเสนอหัวข้อโครงงาน	P	- - -						
	A	———						
2. ศึกษาค้นคว้าหาเก็บรวบรวมข้อมูล	P	- - - - - - -						
	A	———						
3. วิเคราะห์ข้อมูล	P	- - - - - - -						
	A	———	———					
4. ออกแบบระบบ	P		- - - - - -					
	A		———	———				
5. พัฒนาและติดตั้ง	P			- - - - -				
	A			———	———			
6. ทดสอบการทำงาน	P				- - - -			
	A				———	———		
7. ปรับปรุงแก้ไขข้อผิดพลาด	P					- - -		
	A					———		
8. สอบโครงงาน	P					- -		
	A					———		
9. ส่งระบบพร้อมรูปเล่มฉบับสมบูรณ์	P						- - - - -	
	A						———	

- - - - - - แสดงแผนการดำเนินงาน

——— แสดงการดำเนินงานจริง

3.3 ศึกษาและรวบรวมข้อมูลเกี่ยวกับ Sensor วัดกระแสไฟฟ้า

เนื่องจากตัว CT-Sensor ให้สัญญาณออกมาเป็นกระแส ไม่สามารถนำไปต่อกับ MCU โดยตรง ต้องแปลงให้เป็นแรงดันก่อนใช้หลักการ V= I x R ต่อ R Burden เข้ากับเซ็นเซอร์ แล้วให้ MCU วัดแรงดันที่ตกคร่อม R Burden อีกที แต่สัญญาณหลัง R Burden ยังเป็นสัญญาณ AC อยู่ แต่ MCU รับสัญญาณไฟ DC ที่ 0-VCC ดังนั้นต้องยกระดับขึ้นไป 2.5 V จากวงจร R divider



รูปที่ 3.1 วงจรแปลงสัญญาณกระแสเป็นแรงดัน

คำนวณหาค่า R Burden

เนื่องจากตัว CT-Sensor ให้สัญญาณออกมาเป็นกระแส เราต้องคำนวณค่า R Burden เพื่อให้ได้แรงดันตกคร่อม R ที่มีแอมพลิจูดที่เหมาะสมกับพอร์ต ADC ของ MCU เช่นในบอร์ด Arduino ต้องคำนวณค่า R Burden ให้สัญญาณออกมามีแอมพลิจูดไม่เกิน 5V

1) กำหนดย่านการวัดเราต้องทราบก่อนว่าต้องการวัดกระแสในย่านเท่าไหร่หรือโหลดกินกระแสเท่าไหร่ ถ้ากำหนดย่านสูงเกินไป สัญญาณที่ออกมามีขนาดเล็ก จะไม่เห็นความแตกต่างของสัญญาณมากนัก ในตัวอย่างนี้กำหนดที่ 100 A คือที่เซ็นเซอร์สามารถวัดได้สูงสุด

2) หาค่ากระแสสูงสุดในฝั่ง Primary (Primary peak-current) โดยคูณค่ากระแส RMS ด้วย √2

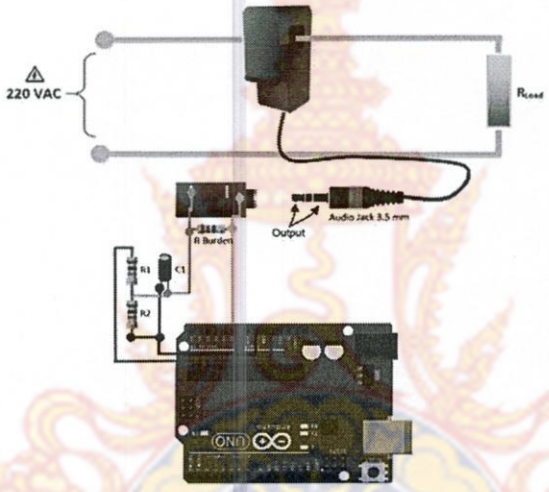
Primary peak-current = RMS current × √2 = 100 A × 1.414 = 141.4A

3) หาค่ากระแสสูงสุดในฝั่ง Secondary (Secondary peak-current) โดยนำค่า Primary peak-current ไปหารจำนวนรอบของ CT ในรุ่นนี้คือ 2000 Secondary peak-current = Primary peak-current / no. of turns = 141.4 A / 2000 = 0.0707A

4) หาค่า R Burden เมื่อ CT วัดกระแสได้สูงสุด ค่าแรงดังสุงสุดที่ผ่าน R Burden จะต้องไม่เกินไฟแรงดันอ้างอิงของพอร์ต ADC (analog reference voltage (AREF)) ของ MCU ที่ใช้ โดยนำไปหาร 2 ก่อน ถ้าใช้บอร์ด Arduino UNO R3 แรงดัน AREF คือ 5V : AREF /2 =5/2=2.5V

ดังนั้นจะหา ค่าความต้านทาน R Burden ในอุดมคติจะหาได้จาก Ideal burden resistance = (AREF/2) / Secondary peak-current = 2.5 V / 0.0707 A = 35.4 Ω แต่ค่า R ทั่วไปที่ใกล้เคียงที่สุดนั้นไม่ใช่ 35 Ω ดังนั้นค่าที่ใกล้เคียงที่สุด คือ 33 Ω

5) ยกระดับแรงดันสัญญาณ ด้วยวิธี DC Bias ตอนนี้ถ้าเราต่อ CT เข้ากับ R burden และทำการวัดกระแส สัญญาณที่ออกมาจาก CT จะยังเป็นสัญญาณ AC อยู่ ดังนั้นถ้าต่อเข้ากับ MCU ที่วัดสัญญาณที่วัดสัญญาณ DC เท่านั้น ต้องยกระดับสัญญาณ AC เป็นไฟ DC โดยจะยกไป AREF/2 จากวงจรแบ่งแรงดัน (Voltage Divider) จาก R1 และ R2 จากสูตร Vout = Vin x R2 /(R1 + R2) เช่น ถ้าใช้บอร์ด Arduino Uno DC Bias ขึ้นไป 2.5 V จึงเลือกใช้ค่า R1, R2 ที่ 100k Ω เท่ากัน ส่วน Capacitor C1 ทำหน้าเป็น Filter noise แนะนำเป็น 10 uF



รูปที่ 3.2 วงจรใช้งาน Non-Invasive Current Sensor - (100A Max) กับ Arduino

ตัวอย่างโค้ดโปรแกรม

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor ems1;                    // Create an instance
void setup()
{
  Serial.begin(9600);
 ems1.current(1, 60.6060606060606);            // Current: input pin, calibration.
}
void loop()
{
  double Irms = ems1.calcIrms(1480);  // Calculate Irms only
```

```
    Serial.print(Irms*230.0);          // Apparent power
    Serial.print(" ");
    Serial.println(Irms);              // Irms
  }
```

อธิบายคำสั่งโค้ดส่วนต่างๆ

```
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor ems1;                    // Create an instance
```

ดึงคำสั่งจาก Library EmonLib มาใช้ โดยใช้คำสั่ง #include "EmonLib.h" จากนั้นประกาศ Instance ชื่อ ems1

```
  void setup()
  {
    Serial.begin(9600);
   ems1.current(1, 60.6060606060606);          // Current: input pin, calibration.
  }
```

ในฟังชั่น Setup ก่อนที่จะเริ่มอ่านค่าจากเซ็นเซอร์ ให้เปิดใช้งาน Hardware Serial กำหนด baud rate เป็น 9600 bps กำหนดค่าให้ฟังชั่น current โดยประกอบด้วย ชื่อขาอนาล็อก และค่า calibrate คำนวณได้จาก CT Ratio / Burden resistance = (100A / 0.05A) / 33 Ohms = 60.6060606060606 จะสังเกตได้ว่ายิ่งใส่ค่า calibrate มาก ความแม่นยำในการวัดจะละเอียดขึ้น

```
  void loop()
  {
    double Irms = ems1.calcIrms(1480);  // Calculate Irms
    Serial.println(Irms);               // Irms
  }
```

จากโค้ดจะเห็นได้ว่าในฟังก์ชั่น Loop วนอ่านสัญญาณอนาล็อกจากเซ็นเซอร์โดยใช้ฟังชั่น calIrms โดยใส่พารามิเตอร์เป็น จำนวนการอ่าน และเก็บไว้ในตัวแปร Irms จากนั้นส่งค่าตัวแปรแสดงผ่านพอร์ต Serial

3.4 ศึกษาและรวบรวมข้อมูลเกี่ยวกับ ESP-02

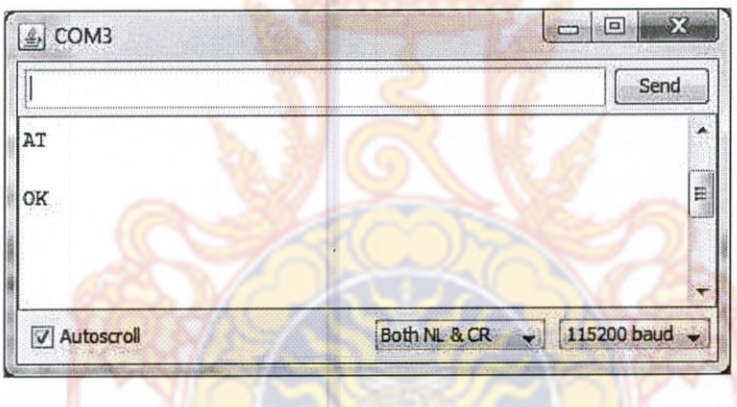
จากการศึกษาโมดูลสื่อสารไร้สาย ESP8266 จะเห็นได้ว่าบางรุ่นนั้นจะใช้เสาอากาศแบบลายวงจรและบางรุ่นใช้เสาอากาศแบบภายนอก ESP8266 รุ่นที่ใช้เสาอากาศภายนอกได้แก่ ESP-02 , ESP-05 และESP-07 จากการทดลอง ESP-05 จะมีปัญหาเรื่องของการอัพเดทเฟิร์มแวร์เพราะต้องกรีดลายวงจรก่อนการอัพเดทเฟิร์มแวร์และ ESP-07 นั้นมีขนาด PCB ที่ใหญ่เกินไปและมีจำนวนขาสำหรับการต่อใช้งานที่มากเกินความจำเป็น ดังนั้นจึงเลือก ESP-02 ในการทำงาน ก่อนการใช้งาน ESP-02 จำเป็นต้องมีการทดสอบการทำงานดังต่อไปนี้

การต่อวงจรจากบอร์ด ESP-02 ไปบอร์ด Arduino

- Vcc-3.3V
- Gnd-Gnd
- CH_PD-3.3V
- TX-TX(ขา 1)
- Rx-RX(ขา 0)

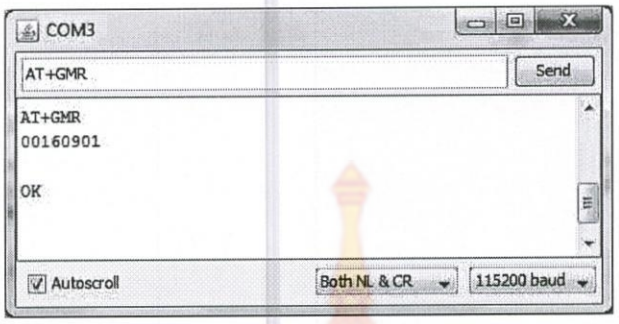
การทดลองขั้นที่ 1 สั่งงานผ่านทาง AT Command

ในตัวอย่างนี้ เราจะใช้ Arduino uno เป็นตัวกลางในการติดต่อ ผ่านทางขา RX , TX เพื่อสั่งงาน AT Command บางครั้งอาจจะมีโปรแกรมใน Arduino uno ที่เราทำไว้ที่จองขา RX , TX เพื่อความแน่ใจ เราจึงอัพโปรแกรมไฟกระพริบลงไปก่อน จากนั้นเปิด Serial Monitor ขึ้นมา ตั้งค่า baud rate 115200 และปรับช่องในรูปให้เป็น Both NL&CR จากนั้นก็พร้อมแล้ว พิมพ์คำสั่ง AT แล้วกด Enter ก็จะเห็นผลลัพธ์จาก ESP-02 ตอบกลับมาว่า OK เป็นอันว่าเชื่อมต่อสำเร็จ



รูปที่ 3.3 การทดสอบคำสั่ง AT Command

คราวนี้เรามาลองทดสอบคำสั่ง AT Command ที่มีให้เราใช้งานใน ESP8266 Wi-Fi Module โดยอ้างอิงจาก เว็บไซต์http://www.electrodragon.com/w/Wi07c ลองดูที่คำสั่ง AT+GMR เป็นคำสั่งเช็คเวอร์ชัน เมื่อพิมพ์แล้วจะได้ผลลัพธ์ออกมาดังนี้ จากรูปจะเห็นว่าเวอร์ชัน Firmware เป็น 0016091 หรือเวอร์ชัน 0.91 ตอนนี้มีเวอร์ชัน 0.9.2.2 ซึ่งเราสามารถเลือกอัพเดทเวอร์ชัน Firmware ได้ เมื่ออัพเดทแล้วจะเป็น 0018000902



รูปที่ 3.4 การตรวจสอบเฟิร์มแวร์ของ ESP8266

คราวนี้เราอยากสั่งงาน ESP8266 ให้ทำอะไร ก็พิมพ์คำสั่ง AT Command ลงไป โดยอ้างอิงจากคู่มือ AT Command หลายคนอาจจะไม่สะดวกหรือไม่เข้าใจในคำสั่ง AT Command ผมได้ทำโปรแกรมช่วยพิมพ์คำสั่ง สำหรับ ESP8266 โดยเฉพาะมาไว้ให้แล้ว โปรแกรมนี้จะแสดงคำสั่ง AT Command วิธีใช้ และคำอธิบายให้ อำนวยความสะดวกในการทดลอง ESP8266 ปิดโปรแกรม Serial Monitor ของ Arduino แล้วเปิดโปรแกรมนี้มาทดลองได้เลย โดยในตัวอย่างนี้ Arduino ต่อกับ COM3 เลือก Board Rae 115200 แล้วกดเชื่อมต่อ จากนั้นก็พิมพ์ AT Command ก็จะขึ้นคำสั่งที่เกี่ยวข้อง เมื่อเลื่อนลงมาก็จะเห็นคำอธิบายของคำสั่งต่างๆ ในตัวอย่างเราจะมาตรวจสอบเวอร์ชันโดยใช้คำสั่ง AT+GMR เหมือนเดิม เมื่อต้องการล้างข้อความผลลัพธ์ใน TextBox เลือกข้อความแล้วกด delete หรือพิมพ์คำสั่ง cls แล้วกด enter ในช่องคำสั่งก็ได้

รูปที่ 3.5 การตรวจสอบเฟิร์มแวร์ของ ESP-02

ขั้นแรก ทดสอบตั้งโหมด Sta+AP

1) ทำการรีเซต ESP-02 โดยพิมพ์ AT+RST

2) เซตเลือกโหมด Wi-Fi Module ESP8266 สามารถตั้งโหมดได้ 3 โหมดโดยใช้คำสั่ง AT+CWMODE

AT+CWMODE : ตั้งค่า Wifi Mode : AT+CWMODE=<โหมด> : เช็คว่าอยู่โหมดไหน AT+CWMODE? : 1 = โหมด Sta 2 = โหมด AP 3 = โหมด Sta และ AP ปกติค่าเริ่มต้นจะเป็นโหมด Sta สำหรับ Router , AP เป็นโหมดปกติสำหรับ Devices เราต้องการตั้งเป็นโหมด Sta+AP แสดงว่าเป็นโหมด 3 ให้พิมพ์ AT+CWMODE=3

รูปที่ 3.6 การเซตเลือกโหมด Wi-Fi Module ESP-02

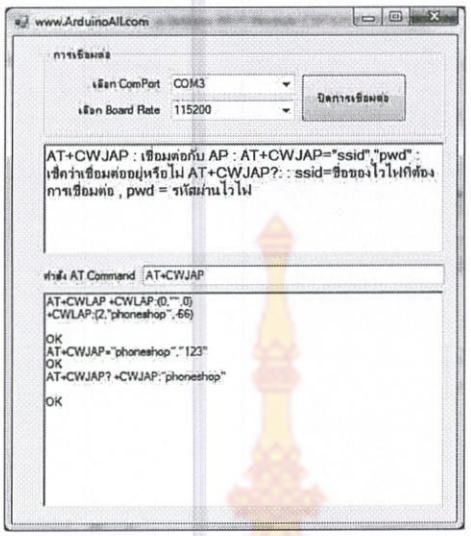
ขั้นที่ 2 เชื่อมต่อกับ Wi-Fi Router

1) ค้นหาเครือข่าย Wi-Fi ที่ใช้ได้ พิมพ์ AT+CWLAP รอซักครู่ ก็จะเห็นรายชื่อ wifi ที่อยู่รอบๆ ตัวเราปรากฏขึ้นมา

2) ทำการเชื่อมต่อกับ Wi-Fi Router ที่ต้องการ โดยพิมพ์ AT+CWJAP="ssid","pwd" ในกรณีนี้เชื่อมต่อกับ phoneshop รหัส 123 ก็พิมพ์ AT+CWJAP="phoneshop","123" ก็จะขึ้นว่า OK

3) ตรวจสอบผลการเชื่อมต่อโดยพิมพ์ AT+CWJAP?

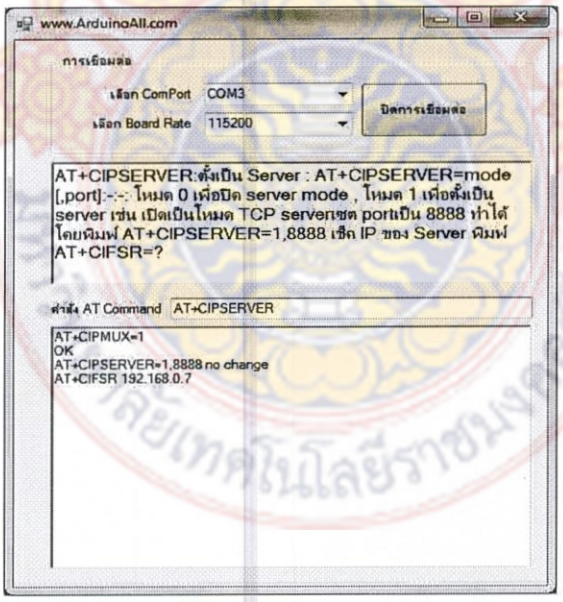
เมื่อตั้งค่าเรียบร้อยแล้ว เปิดอุปกรณ์ขึ้นมาใหม่ wifi ก็จะเชื่อมต่อกับเครือข่ายที่เราตั้งไว้อัตโนมัติทุกครั้ง



รูปที่ 3.7 การเชื่อมต่อกับ Wi-Fi Router

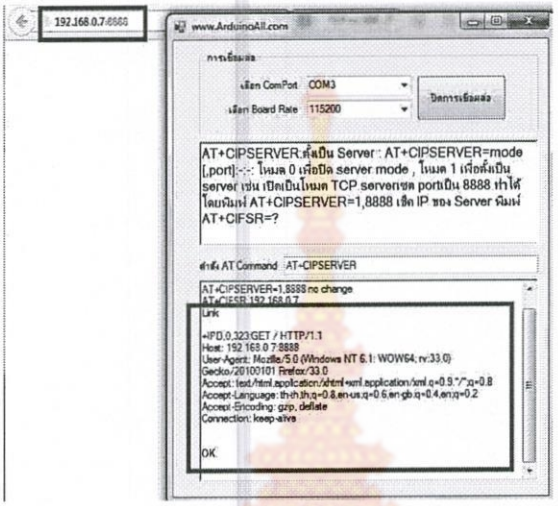
ขั้นที่ 3 ทำ Wi-Fi Module ESP-02 เป็น TCP Server

- AT+CIPMUX=1 เปิดโหมดการเชื่อมต่อแบบหลายจุด
- AT+CIPSERVER=1,8888 Setup TCP server, ช่อง Port 8888
- AT+CIFSR ตรวจสอบไอพี



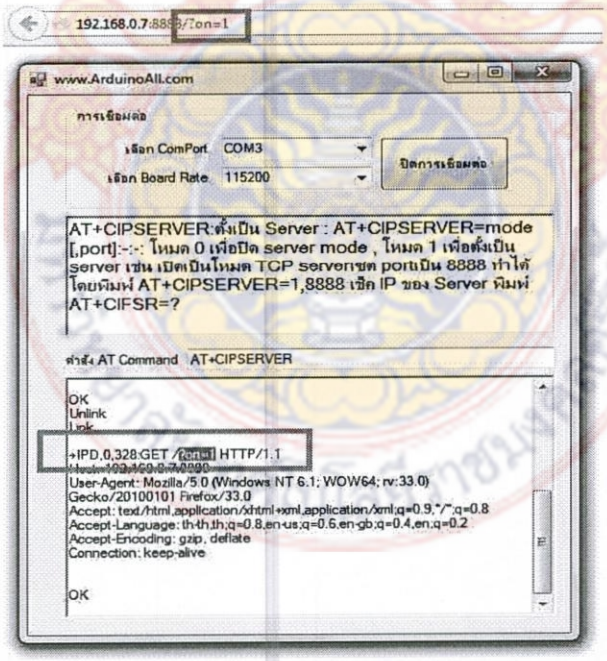
รูปที่ 3.8 การทำ Wi-Fi Module ESP-02 เป็น TCP Server

จากนั้นเปิดเว็บขึ้นมา พิมพ์ ip และพอร์ท แล้วสังเกตผลลัพธ์ จะเห็นว่าเราสามารถส่งค่าจากเว็บมายัง Wi-Fi Server ของเราได้แล้ว



รูปที่ 3.9 ผลลัพธ์การส่งค่าจากเว็บมายัง Wi-Fi Server

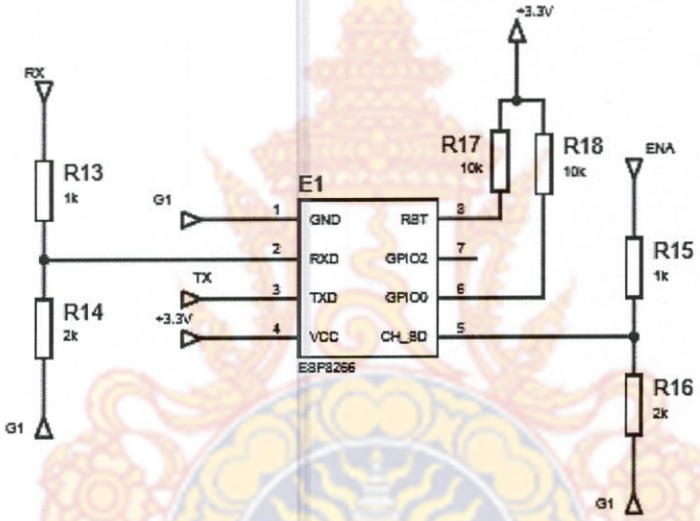
เราสามารถส่งค่าพารามิเตอร์ไปกับ url เพื่อควบคุมให้ Arduino Server ของเราทำงานตามที่ต้องการ



รูปที่ 3.10 การส่งค่าพารามิเตอร์ไปกับ url

จากการทดลองนี้กระทำเพื่อตรวจสอบตัวโมดูล ESP-02 ว่าพร้อมใช้งานหรือไม่ เมื่อตรวจสอบผ่านก็ทำการอัพเดทเฟิร์มแวร์ต่อไป หลังจากนั้นก็ทำการปรับเปลี่ยนการต่อวงจร ESP-02 ดังนี้

- Vcc-3.3V
- Gnd-Gnd
- CH_PD-(ขา 9)
- TX-RX(ขา 10)
- Rx-TX(ขา 11)
- GPIO0-3.3V
- RST-3.3V



รูปที่ 3.11 การต่อวงจรของ ESP-02

จากการต่อใช้งานจริง ESP-02 จะใช้ไฟ 3.3 V จากบอร์ด Arduino ไม่ได้ เนื่องจาก กระแสในการส่งข้อมูลไม่เพียงพอเพราะภาคจ่ายไฟของ Arduino มีกระแสเพียง 50 mA และไม่ควรต่อกับภาคจ่ายไฟ 5 V เพราะจะทำให้ ESP-02 มีประสิทธิภาพในการทำงานน้อยลงและเกิดความเสียหายได้ ส่วนการต่อขา RX ของ ESP-02 ไปยังบอร์ด Arduino ต้องมีการต่อ voltage divider เพื่อลดแรงดันของบอร์ด Arduino ขณะที่มีการส่งข้อมูลออกมาจาก 5 V เหลือเพียงแค่ 3.3 V

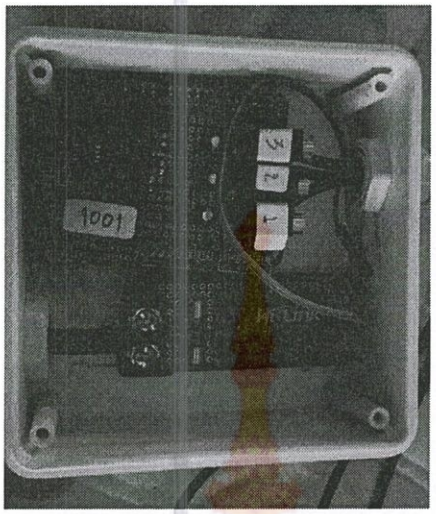
Mode	Min	Typ	Max	Unit
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm		215		mA
Transmit 802.11b, CCK 11Mbps, POUT=+18.5dBm		197		mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm		145		mA
Transmit 802.11n, MCS7, POUT=+14dBm		135		mA
Receive 802.11b, packet length=1024 byte, -80dBm		60		mA
Receive 802.11g, packet length=1024 byte, -70dBm		60		mA
Receive 802.11n, packet length=1024 byte, -65dBm		62		mA
Standby		0.9		mA
Deep sleep		10		uA
Power save mode DTIM 1		1.2		mA
Power save mode DTIM 3		0.86		mA
Total shutdown		0.5		uA

รูปที่ 3.12 ตารางแสดงค่ากระแสที่ต้องใช้ในแต่ละโหมด

การเขียนโค้ดคำสั่งควบคุม ESP-02 ควรมีการหน่วงเวลาที่เหมาะสมเพื่อให้คำสั่งได้รันเสร็จสิ้นมิฉะนั้นแล้วจะทำให้เกิดการความผิดพลาดในการส่งข้อมูลได้ เช่นการเกิด Busy... , Busy p... และBusy s...

```
--AT+CIPSTATUS
AT+CIPSTATUS
busy p...
□o2ก□not connect to server!!!
Wake up
--AT+CIPSTATUS
AT+CIPSTATUS
busy p...
```

รูปที่ 3.13 การเกิด busy p...

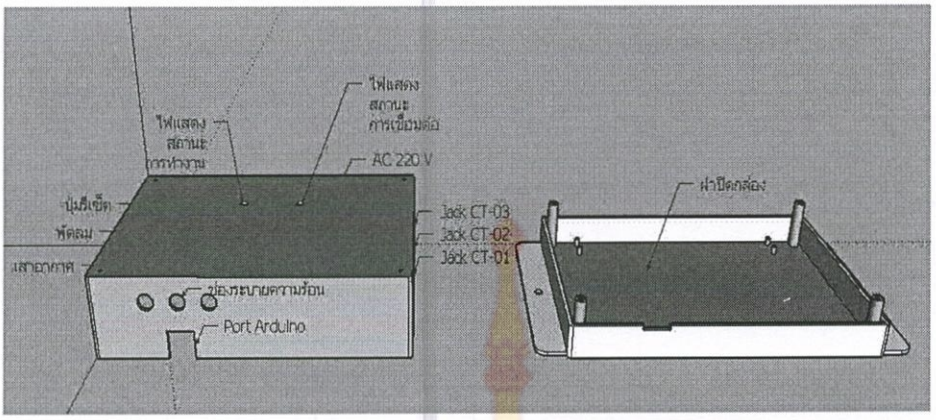


รูปที่ 3.14 ทดลองใช้ module esp8266 ส่งค่ากระแส

3.5 การออกแบบกล่องอเนกประสงค์

การออกแบบโครงสร้างของกล่องอเนกประสงค์สำหรับใส่แผงวงจรและอุปกรณ์ต่างๆ ภายในนั้นต้องนำกล่องอเนกประสงค์มาทำการวัดขนาดและใช้โปรแกรม SketchUp 2015 ในการสร้างแบบแปลนของกล่องขึ้นมา และทำการกำหนดจุดต่างๆ ที่จะทำการจัดวางแผงวงจรและอุปกรณ์ต่างๆ ภายในก่อนทำการเจาะ เพราะการออกแบบก่อนจะทำให้ไม่เสียเวลาในการการดำเนินการ และลดการสูญเสียอุปกรณ์รวมถึงค่าใช้จ่ายที่เกิดขึ้น ดังรูปที่ 3.1 และ 3.2

รูปที่ 3.15 กล่องอเนกประสงค์สำหรับใส่แผงวงจร

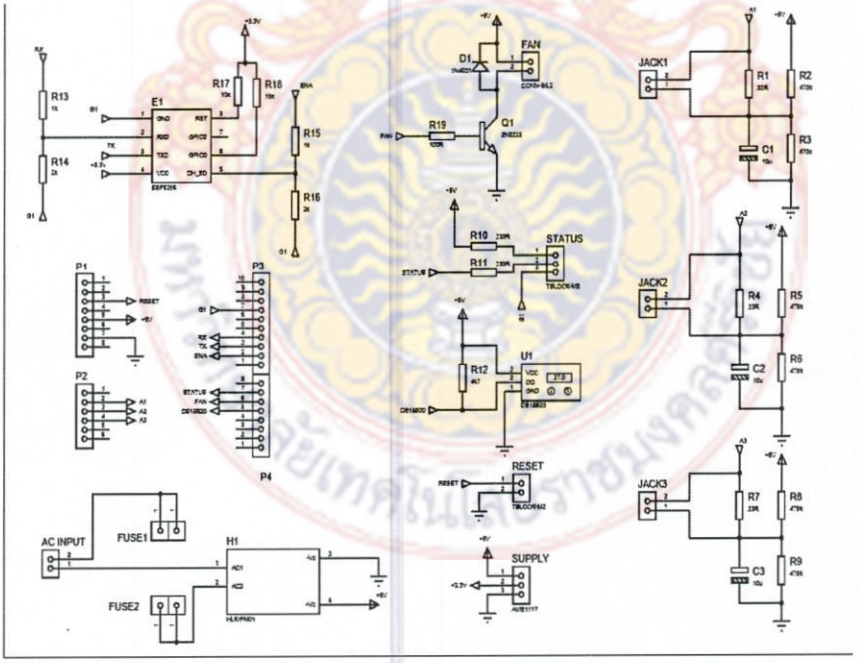


รูปที่ 3.16 แบบแปลนของกล่องอเนกประสงค์ที่สร้างโดยโปรแกรม SketchUp 2015

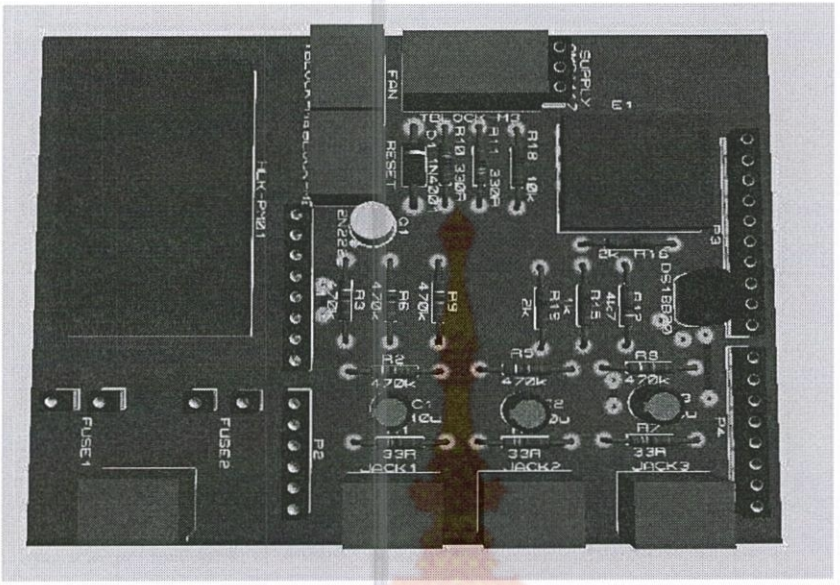
3.6 การออกแบบโครงสร้างและการจัดทำแผงวงจร

การออกแบบโครงสร้างของแผงวงจรนั้นจำเป็นต้องมีความรู้ความเข้าใจเกี่ยวกับตัวอุปกรณ์และลายวงจร โดยการออกแบบจะใช้โปรแกรม Proteus ในการออกแบบได้จากการศึกษารวบรวมข้อมูลจากหัวข้อที่ 3.3 และ 3.4

1) ออกแบบลายวงจรด้วยโปรแกรม Proteus

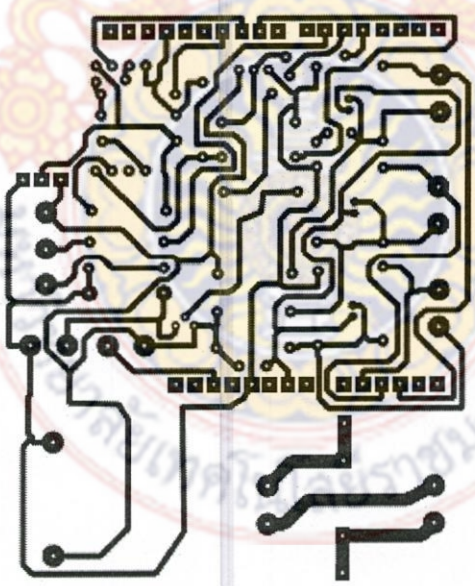


รูปที่ 3.17 แผนผังวงจรใช้โปรแกรม Proteus ในการออกแบบ



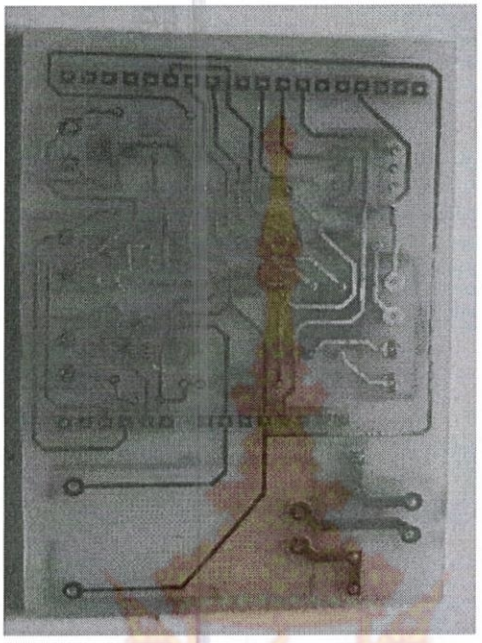
รูปที่ 3.18 แบบจำลองแผงวงจรที่ใช้โปรแกรม Proteus ในการออกแบบ

2) ปริ๊นลายวงจรที่ทำออกมา



รูปที่ 3.19 PCB ลายวงจร

3) แผ่นปริ้นท์ที่กัดเสร็จแล้วมาล้างออกด้วยน้ำสะอาดก่อนจะนำไปทำการเจาะรูเพื่อใส่อุปกรณ์



รูปที่ 3.20 แผ่นปริ๊นท์ที่ทำการเจาะรูเสร็จเรียบร้อย

4) ทำการต่อทดสอบเพื่อตรวจเช็คความพร้อมของอุปกรณ์

รูปที่ 3.21 ทดสอบความพร้อมของอุปกรณ์

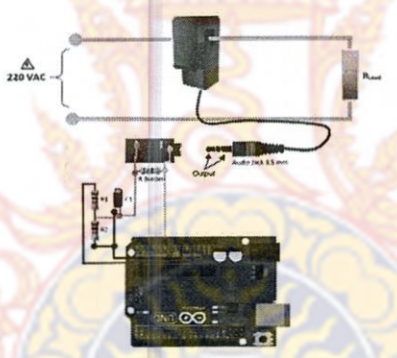
บทที่ 4
ผลการดำเนินงานและการวิเคราะห์

ในการออกแบบและสร้างอุปกรณ์วัดการใช้กำลังงานไฟฟ้าแบบไร้สาย จะต้องหาผลการทดลองที่ดีที่สุด เพื่อนำมาเป็นต้นแบบในการดำเนินการ ให้มีความถูกต้องแม่นยำเกิดความผิดพลาดน้อยที่สุด โดยใช้เซนเซอร์วัดกระแสไฟฟ้าแบบ Current Transformer ในการตรวจวัดค่ากระแสไฟฟ้า และจะใช้โมดูลสื่อสารไร้สาย ESP-02 ในการส่งค่าที่วัดได้ผ่านสัญญาณไวไฟ ซึ่งจะทำการทดลองการส่งข้อมูลของค่ากระแสไฟฟ้าที่วัดได้ผ่านสัญญาณไวไฟ เพื่อการใช้งานที่มีประสิทธิภาพมากที่สุด

4.1 ขั้นตอนการทำงาน

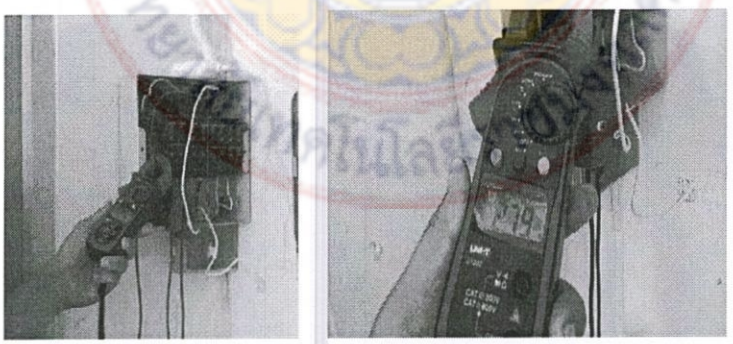
การทดลองระบบส่งข้อมูล โดยใช้โมดูลสื่อสารไร้สาย ESP-02 ในการทดลองส่งข้อมูลค่ากระแสที่วัดได้เข้าสู่เว็บฐานข้อมูล

4.1.1 ต่อวงจงตามรูป ในบทที่ 3



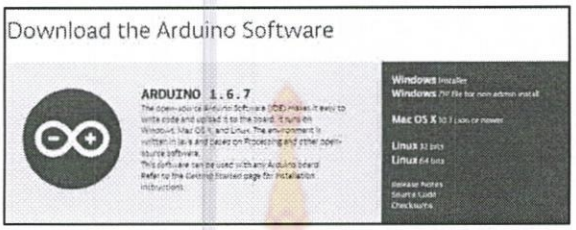
รูปที่ 4.1 แสดงการต่อวงจร

- เมื่อต่อแล้ว ทำการวัดโดยเปรียบเทียบกับเครื่องมือวัดมาตราฐาน



รูปที่ 4.2 แสดงการต่อวงจรตามจริง

4.1.2 ติดตั้ง Arduino IDE เวอร์ชั้น 1.6.7 หรือ ใหม่กว่า โดย Download ตัวติดตั้งได้จาก http://www.arduino.cc/en/main/software



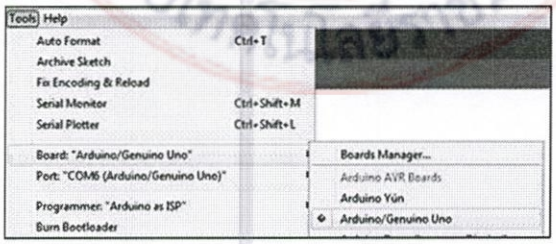
รูปที่ 4.3 แสดงไอคอนการดาวน์โหลดโปรแกรม Arduino IDE เวอร์ชั้น 1.6.7

-เมื่อติดตั้งโปรแกรม Arduino เสร็จแล้วจะได้หน้าต่างดังรูปนี้

รูปที่ 4.4 แสดงหน้าต่างของโปรแกรม Arduino IDE เวอร์ชั้น 1.6.7

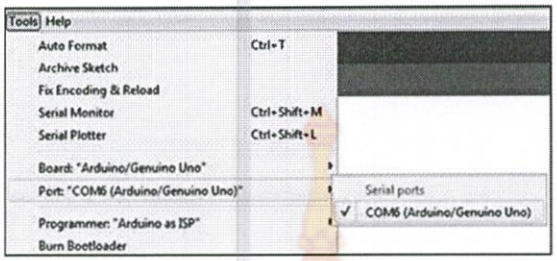
4.1.3 เช็คพอร์ตการเชื่อมต่อดูว่า Arduino นั้นได้เชื่อมต่อเข้ากับโปรแกรมแล้ว ไปที่ Tools ดูหัวข้อ Board และ Port ว่าเป็นชื่อ Arduino ตัวที่เราได้เชื่อมต่อไว้หรือไม่

-คลิกเลือกรุ่นบอร์ด arduino/uno



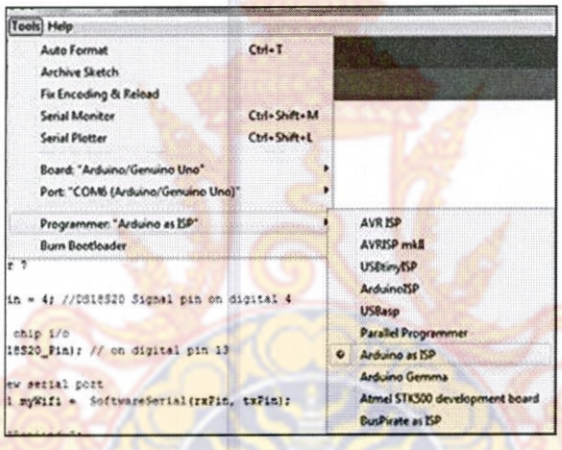
รูปที่ 4.5 แสดงการเลือกรุ่นบอร์ด arduino/uno

-คลิกเลือก Port



รูปที่ 4.6 แสดงการเลือก Port

-คลิกเลือก Arduino as ISP

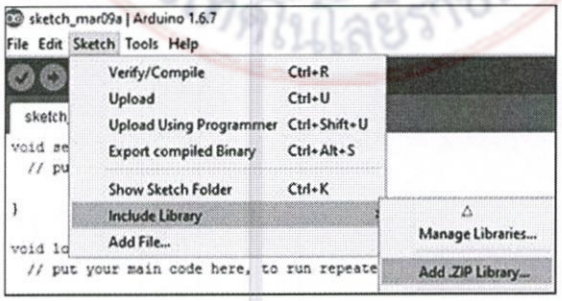


รูปที่ 4.7 แสดงการเลือก Arduino as ISP

4.1.4 โหลด Library EmonLib ที่ https://github.com/openenergymonitor/EmonLib

-เมื่อดาวโหลเสร็จนำ Library ไปติดตั้งที่ C:\Program Files\Arduino\libraries

-ทำการเพิ่ม Library Code โดยกดที่ Sketch>>Include Library>>Add .ZIP Library...



รูปที่ 4.8 แสดงการเพิ่ม Library Code

-คลิกเลือกfolder และเลือกfileตามที่เราต้องการ

-ทำการเขียน code โปรแกรมใน Arduino จะได้ดังนี้

```
#include <SoftwareSerial.h>
#include <OneWire.h>
#define pinEN 9
#define rxPin 10
#define txPin 11
#define  ID  "1002"
#define Status 6
#define Buffer 7
int DS18S20_Pin = 4; //DS18S20 Signal pin on digital 4
//Temperature chip i/o
//OneWire ds(DS18S20_Pin); // on digital pin 13
// set up a new serial port
SoftwareSerial myWifi =  SoftwareSerial(rxPin, txPin);
String ssid = "Techcom08";
String pass = "techcom1995";
//String serverip = "siamcyber.dyndns.info";
String serverip = "172.16.9.179";
String path = "/googleChartAPI/lineGrapAPI/apit.php";
#define TIMEOUT     10000 // mS
#define CONTINUE    false
#define HALT        true
#define ECHO_COMMANDS // Un-comment to echo AT+ commands
to serial monitor
// ----  current -----
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon11;                  // Create an instance
EnergyMonitor emon12;                  // Create an instance
EnergyMonitor emon13;                  // Create an instance
// Print error message and loop stop.
void errorHalt(String msg){
Serial.println(msg);
Serial.println("HALT");
while(true){};
```

```
}
// Read characters from WiFi module and echo to serial until keyword occurs or timeout.
boolean echoFind(String keyword)
{
byte current_char   = 0;
byte keyword_length = keyword.length();
// Fail if the target string has not been sent by deadline.
long deadline = millis() + TIMEOUT;
while(millis() < deadline)
{
if (myWifi.available())
{
char ch = myWifi.read();
Serial.write(ch);
if (ch == keyword[current_char])
if (++current_char == keyword_length)
{
Serial.println();
return true;
}
}
}
return false;  // Timed out
}
// Read and echo all available module output.
// (Used when we're indifferent to "OK" vs. "no change" responses or to get around firmware bugs.)
void echoFlush()
{while(myWifi.available()) Serial.write(myWifi.read());}
// Echo module output until 3 newlines encountered.
// (Used when we're indifferent to "OK" vs. "no change" responses.)
void echoSkip()
{
echoFind("\n");       // Search for nl at end of command echo
echoFind("\n");       // Search for 2nd nl at end of response.
```

```
echoFind("\n");       // Search for 3rd nl at end of blank line.
}
// Send a command to the module and wait for acknowledgement
string
// (or flush module output if no ack specified).
// Echoes all data received to the serial monitor.
boolean echoCommand(String cmd, String ack, boolean halt_on_fail)
{
myWifi.println(cmd);
#ifdef ECHO_COMMANDS
Serial.print("--"); Serial.println(cmd);
#endif
// If no ack response specified, skip all available module output.
 if (ack == "")
echoSkip();
else
// Otherwise wait for ack.
if (!echoFind(ack))          // timed out waiting for ack string
if (halt_on_fail)
errorHalt(cmd+" failed");// Critical failure halt.
else
return false;          // Let the caller handle it.
return true;                 // ack blank or ack found
}
// Connect to the specified wireless network.
boolean connectWiFi()
{
String cmd = "AT+CWJAP=" + String("\"") + ssid + String("\"") +
String(",")+String("\"") + pass +String("\"");
if (echoCommand(cmd, "OK", CONTINUE)) // Join Access Point
{
Serial.println("Connected to WiFi.");
return true;
}
else
{
```

```
Serial.println("Connection to WiFi failed.");
return false;
}
}
boolean cipmux0()
{
myWifi.println("AT+CIPMUX=0");
Serial.print("Multipleconnect= ");
if (myWifi.find("OK"))
{
Serial.print(" Single mode");
return true;
}
else
{
return false;
}
}
//-------------------------------------------------------------------
boolean cipmode0()
{
myWifi.println("AT+CIPMODE=0");
if (myWifi.find("OK"))
{
return true;
}
else
{
return false;
}
}
//-------------------------------------------------------------------
// ******** SETUP ********
void setup()
{
pinMode(Status, OUTPUT);
```

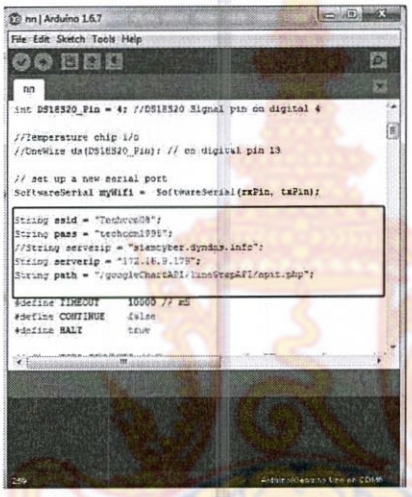
```
pinMode(Buffer, OUTPUT);
pinMode(pinEN, OUTPUT);
pinMode(rxPin, INPUT);
pinMode(txPin, OUTPUT);
digitalWrite(pinEN, HIGH);
digitalWrite(Status, HIGH);  // turn the LED Status on
 digitalWrite(Buffer, HIGH);  // turn the LED Buffer on
Serial.begin(9600);         // Communication with PC monitor via USB
myWifi.begin(9600);         // Communication with ESP8266 via 5V/3.3V
level shifter
myWifi.setTimeout(TIMEOUT);
Serial.println("\r < Electric energy measuring devices With the online
system >");
 delay(2000);
echoCommand("AT+RST", "OK", CONTINUE);      // Reset & test if the
module is ready
Serial.println("Module is ready.");
echoCommand("AT+GMR", "OK", CONTINUE);  // Retrieves the firmware
ID (version number) of the module.
echoCommand("AT+CWMODE?","OK", CONTINUE);// Get module access
mode.
 echoCommand("AT+CWMODE=1", "", HALT);    // Station mode
echoCommand("AT+CWQAP", "OK", CONTINUE);          // Quit Access
Point
//connect to the wifi
boolean connection_established = false;
for(int i=0;i<5;i++)
{
if(connectWiFi())
{
connection_established = true;
break;
}
}
if (!connection_established) errorHalt("Connection failed");
delay(5000);
```

```
echoCommand("AT+CIFSR", "", HALT);                // Echo IP address.
(Firmware bug - should return "OK".)
if(!cipmux0()) errorHalt("cipmux0 failed");
delay(250);
if(!cipmode0()) errorHalt("cipmode0 failed");
delay(250);
digitalWrite(Buffer, LOW);  //turn the LED Buffer off when set status
module wifi finish.
// current
//emon11.current(1, 60.6);          // Current: input pin, calibration. 33
Ohms
//emon12.current(2, 60.6);          // Current: input pin, calibration. 33
Ohms
emon13.current(3, 60.6);          // Current: input pin, calibration. 33
Ohms
}
// ******** LOOP ********
void loop()
{
delay(100); //just here to slow down the output so it is easier to read
//double Irms1 = emon11.calcIrms(1480);  // Calculate Irms only
//double Irms2 = emon12.calcIrms(1480);  // Calculate Irms only
double Irms3 = emon13.calcIrms(1480);  // Calculate Irms only
// Establish TCP connection
st:
String cmd = "AT+CIPSTART=\"TCP\",\"";  //make this command:
                        AT+CPISTART="TCP","146.227.57.195",80
cmd += serverip;
cmd += "\",80";
myWifi.println(cmd);  //send command to device
delay(2000);  //wait a little while for 'Linked' response - this makes a
difference
if(myWifi.find("Linked"))    //message returned when connection
established WEAK SPOT!! DOESN'T ALWAYS CONNECT
{
Serial.print("Connected to server at ");  //debug message
```

```
Serial.println(serverip);
}
else
{
Serial.println("'Linked' response not received");  //weak spot! Need to recover elegantly
goto st;
}
// Build HTTP request.
cmd = "GET "+path;
cmd +="/?id="+String(ID);
cmd += "&v1="+String(Irms3)+" HTTP/1.0\r\n\r\n";
delay(2500);
myWifi.print("AT+CIPSEND=");
myWifi.println(cmd.length());  //esp8 2 6 6  needs to know message length of incoming message - .length provides this
delay(1500);
if(myWifi.find(">"))  //prompt offered by esp8266
{
Serial.println("sent ok");  //a debug message
myWifi.println(cmd);  //this is our http GET request
Serial.println(cmd);
digitalWrite(Buffer, HIGH); delay(600);
digitalWrite(Buffer, LOW);
}
else
{
digitalWrite(Buffer, HIGH); delay(1000);
myWifi.println("AT+CIPCLOSE");  //doesn't seem to work here?
Serial.println("No '>' prompt received after AT+CPISEND");
digitalWrite(Buffer, LOW); delay(200);
}
myWifi.println("AT+CIPCLOSE");
if(myWifi.find("Unlink"))  //rarely seems to find Unlink? :(
{
Serial.println("Connection Closed Ok...");
```

```
}
else
{
Serial.println("Connection Closed Error !!!");
}
}
```

4.1.5 เปลี่ยน ssid , Password , serverip ,port และ url เป็นของเครื่องที่ใช้งาน

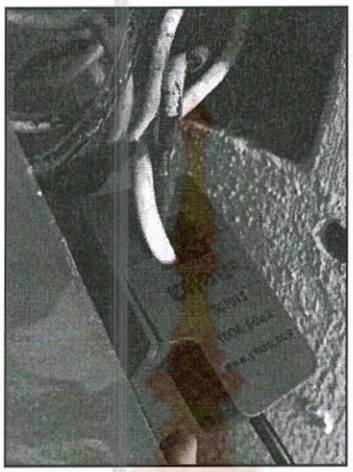


รูปที่ 4.9 แสดงการเปลี่ยน ssid , Password , serverip ,port และ url เป็นของเครื่องที่ใช้งาน

-คลิกเลือกserial moniterเพื่อดูผลการทำงาน จะเป็นรูปแว่นขยายอยู่มุมขวาของโปรแกรม

รูปที่ 4.10 แสดงผลการทำงานด้วย Arduino IDE เวอร์ชั้น 1.6.7

4.1.6 ทำการวัดพลังงานจากสายไฟ



รูปที่ 4.11 แสดงการวัดสายไฟด้วย CT Sensor

4.1.7 แสดงผลการวัดเป็นกราฟผ่านทางหน้าweb โดยใช้ code web program จาก google chart

จากกราฟ ก่อนวัดกระแสอยู่ที่ ≈ 1.5A และเมื่อทำการวัดกราฟแสดงที่ ≈ 11.1A ซึ่งตรงกับค่าที่กัดได้จาก Clamp Meter

รูปที่ 4.12 แสดงการวัดสายไฟด้วย Clamp Meter

รูปที่ 4.13 แสดงผลการวัดด้วยระบบออนไลน์

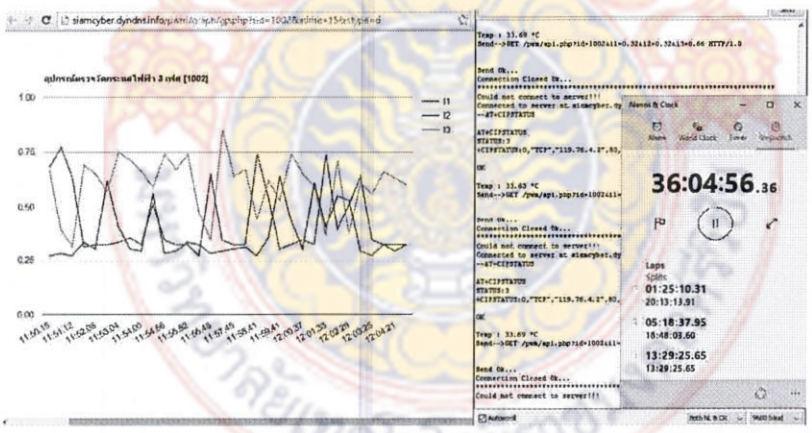
4.1 ตารางเปรียบเทียบการวัดค่าจาก Clamp Meter และ CT sensor

ค่ากระแสไฟฟ้า (A)	ผลการทดลอง		
	การทดลองครั้งที่ 1	การทดลองครั้งที่ 2	การทดลองครั้งที่ 3
Digital Clamp Multimeter	2.79	2.86	1.92
เซนเซอร์วัดกระแสไฟฟ้า	2.61	2.75	1.85

บทที่ 5
สรุปและข้อเสนอแนะ

จากกระบวนการทดลองจำเป็นต้องมีข้อสรุปในการทดลอง เพื่อให้ทราบถึงความสามารถและประสิทธิภาพในการทำงานที่แน่นอนและชัดเจน การสรุปผลการทดลองนี้ทำได้โดยนำผลการทดลองมาหาข้อสรุปถึงจุดเด่น จุดด้อย โดยการทดลองความสามารถและประสิทธิภาพของโมดูลสื่อสารไร้สาย ESP-02 และเซนเซอร์วัดกระแสไฟฟ้าแบบ Current Transformer ซึ่งจะสรุปในส่วนของการทดลองระบบส่งข้อมูล การทดลองระบบการวัดค่ากระแสไฟฟ้า โดยการนำผลการทดลองมาเป็นเกณฑ์ในการพิจารณาหาผลสรุป เปรียบเทียบกับปัญหาและอุปสรรคที่เกิดขึ้นจากการทดลอง การสรุปผลสามารถทำให้เข้าใจถึงประสิทธิภาพและรูปแบบการทำงาน การใช้งานจริง ซึ่งจะทำให้สามารถให้ข้อเสนอแนะและแนวทางในการพัฒนาให้ระบบมีประสิทธิภาพมากยิ่งขึ้น
จากการศึกษาครั้งนี้ผู้ศึกษาได้ทำการศึกษาถึงการทำงานของอุปกรณ์ Hardware และ Software ของเครื่องวัดกระแสไฟฟ้าด้วย CT Sensor และแสดงผลผ่านทางหน้าเว็บ ได้ทำการวัดจากสายไฟหลาย ๆ ที่ทั้งปรากฏค่าและไม่ปรากฏค่าแตกต่างกันไป โดยใช้บอร์ด Arduino เป็นไมโครคอลโทรเลอร์ใช้โปรแกรม Arduino IDE ในการเขียนโปรแกรม และใช้โมดูล Esp 8266 ในการส่งค่าที่วัดได้เข้าสู่ฐานข้อมูลเพื่อแสดงผลผ่านทางหน้าเว็บต่อไป

5.1 สรุปผลการทดลอง



รูปที่ 5.1 การตรวจสอบระบบส่งข้อมูลของโมดูลสื่อสารไร้สาย ESP-02

สามารถใช้งานระบบการวัดค่ากระแสไฟฟ้า โดยใช้เซนเซอร์วัดกระแสไฟฟ้าแบบ Current Transformer ใช้ในการตรวจวัดค่ากระแสไฟฟ้าตั้งแต่ 0-100 แอมแปร์ ซึ่งการวัดค่ากระแสเมื่อเทียบกับ Digital Clamp Multimeter UNI-T UT2002 จะมีค่าผิดพลาดในการวัดไม่เกิน 5 เปอร์เซ็น

5.2 อุปสรรคและปัญหา

5.2.1 จากการทดลองจะเกิด busy p... ของ ESP-02 เนื่องจากการรันคำสั่ง AT Command ไม่สมบูรณ์

5.2.2 กระแสที่ใช้เลี้ยงโมดูลสื่อสารไร้สาย ESP-02 ไม่เพียงพอ เนื่องจากโมดูลสื่อสารไร้สาย ESP-02 ใช้กระแสในการส่งข้อมูลอยู่ที่ 215 มิลลิแอมป์ แต่ภาคจ่ายไฟ 3.3 โวลต์ ของบอร์ด Arduino UNO R3 ให้กระแสเพียง 50 มิลลิแอมป์ ซึ่งไม่เพียงพอในการส่งข้อมูลของโมดูลสื่อสารไร้สาย ESP-02

5.2.3 ค่ากระแสในการวัดที่ไม่ตรงกัน เกิดจากการบัดกรีตะกั่วเกาะอยู่ที่ขาอุปกรณ์เท่านั้นไม่ได้เชื่อมกับลายวงจร

5.3 ข้อเสนอแนะ

จากการศึกษาและการทดลองระบบส่งข้อมูลและระบบการวัดค่ากระแสไฟฟ้า สามารถที่จะนำแนวทางการทำงานของระบบและอุปกรณ์ไปปรับปรุงประสิทธิภาพการทำงานให้มีเสถียรภาพมากขึ้น ในการออกแบบโครงสร้างและโปรแกรมให้เหมาะสม โดยการพัฒนาภาคจ่ายไฟให้มีกระแสที่เพียงพอสำหรับโมดูลสื่อสารไร้สาย ESP-02 ปรับปรุงการใช้บอร์ดให้เหลือเพียงบอร์ดเดียว โดยการรวมวงจรจ่ายไฟและวงจรควบคุมเข้าด้วย ทำให้สะดวกต่อการใช้งานและประหยัดพื้นที่ รวมถึงการใช้ไฟ 220 โวลต์ในการการใช้งานจะสามารถช่วยประหยัดการใช้อแดปเตอร์ ตรวจสอบการบัดกรีอุปกรณ์เพื่อให้แน่ใจว่าอุปกรณ์กับลายวงจรเชื่อมต่อกันอย่างสมบูรณ์และการใช้เซนเซอร์วัดกระแสไฟฟ้าแบบ Current Transformer นั้นจะสามารถทำการวัดกระแสไฟฟ้าโดยการใช้เซนเซอร์ 1 ตัว ต่อสายไลน์ 1 เส้น เพราะถ้าใช้เซนเซอร์วัดกระแสไฟฟ้าต่อคร่อมสายไลน์พร้อมกัน 2 ตัวจะทำให้เกิดสนามแม่เหล็กและทำให้การวัดเกิดค่าผิดพลาดขึ้น

บรรณานุกรม

[1] **"หลายหลากคำถามเกี่ยวกับ CT Current Transformer" Questions about CT - See more.** [ออนไลน์]. เข้าถึงได้จาก :
http://www.engineerfriend.com/2012/articles/หลากหลายคำถามเกี่ยวกับ-c/
(วันที่ค้นหาข้อมูล : 3 กุมภาพันธ์ 2559).

[2] **How to build an Arduino energy monitor** . [ออนไลน์]. เข้าถึงได้จาก :
https://openenergymonitor.org/emon/node/58
(วันที่ค้นหาข้อมูล : 2 กุมภาพันธ์ 2559).

[3] **CT sensors-introduction** . [ออนไลน์]. เข้าถึงได้จาก :
https://openenergymonitor.org/emon/buildingblocks/ct-sensors-introduction
(วันที่ค้นหาข้อมูล : 7 กุมภาพันธ์ 2559).

[4] **CT sensors - Interfacing with an Arduino** . [ออนไลน์]. เข้าถึงได้จาก :
https://openenergymonitor.org/emon/buildingblocks/ct-sensors-interface
(วันที่ค้นหาข้อมูล : 8 กุมภาพันธ์ 2559).

[5] **Introduction to Current Transformers** . [ออนไลน์]. เข้าถึงได้จาก :
https://www.elkor.net/pdfs/AN0305-Current_Transformers.pdf
(วันที่ค้นหาข้อมูล : 14 กุมภาพันธ์ 2559).

[6] **ความรู้เบื้องต้นเกี่ยวกับ Current Sensor (เซ็นเซอร์วัดกระแส)** . [ออนไลน์]. เข้าถึงได้จาก :
http://www.thaieasyelec.com/article-wiki/review-product-article/ความรู้เบื้องต้นเกี่ยวกับ current-sensorเซ็นเซอร์วัดกระแส.html
(วันที่ค้นหาข้อมูล : 2 กุมภาพันธ์ 2559).

[7] **ความรู้เบื้องต้นเกี่ยวกับ Current Sensor (เซ็นเซอร์วัดกระแส)** . [ออนไลน์]. เข้าถึงได้จาก :
http://www.thaieasyelec.com/article-wiki/review-product-article/ตัวอย่างการใช้งาน-current-sensors-เซ็นเซอร์วัดกระแส-ประเภท-current-transformer-sensor.html
(วันที่ค้นหาข้อมูล : 5 กุมภาพันธ์ 2559).

ภาคผนวก ก

การติดตั้ง Arduino IDE และ Library

ก.1 Arduino IDE

ก.1.1 ติดตั้ง Arduino IDE เวอร์ชั้น 1.6.7 หรือ ใหม่กว่า โดย Download ตัวติดตั้งได้จาก http://www.arduino.cc/en/main/software

ก.2 การติดตั้ง Library

ก.2.1 ติดตั้ง Library EmonLib

โหลด Library EmonLib ที่ https://github.com/openenergymonitor/EmonLib

เมื่อดาวโหลเสร็จนำ Library ไปติดตั้งที่ C:\Program Files\Arduino\libraries

ก.2.2 ทำการเพิ่ม Library Code โดยกดที่ Sketch>>Include Library>>Add .ZIP Library

ภาคผนวก ข
โค้ดโปรแกรม

ข.1 โค้ดโปรแกรม

```
#include <SoftwareSerial.h>
#include <OneWire.h>
#define pinEN 9
#define rxPin 10
#define txPin 11
#define  ID  "1002"
#define Status 6
#define Buffer 7
int DS18S20_Pin = 4; //DS18S20 Signal pin on digital 4
//Temperature chip i/o
//OneWire ds(DS18S20_Pin); // on digital pin 13
// set up a new serial port
SoftwareSerial myWifi =  SoftwareSerial(rxPin, txPin);
String ssid = "Techcom08";
String pass = "techcom1995";
//String serverip = "siamcyber.dyndns.info";
String serverip = "172.16.9.179";
String path = "/googleChartAPI/lineGrapAPI/apit.php";
#define TIMEOUT     10000 // mS
#define CONTINUE    false
#define HALT        true
#define ECHO_COMMANDS // Un-comment to echo AT+ commands
to serial monitor
// ----  current -----
#include "EmonLib.h"                   // Include Emon Library
EnergyMonitor emon11;                  // Create an instance
EnergyMonitor emon12;                  // Create an instance
EnergyMonitor emon13;                  // Create an instance
// Print error message and loop stop.
void errorHalt(String msg){
Serial.println(msg);
Serial.println("HALT");
while(true){};
}
```

```
// Read characters from WiFi module and echo to serial until keyword occurs or timeout.
boolean echoFind(String keyword)
{
byte current_char   = 0;
byte keyword_length = keyword.length();
// Fail if the target string has not been sent by deadline.
long deadline = millis() + TIMEOUT;
while(millis() < deadline)
{
if (myWifi.available())
{
char ch = myWifi.read();
Serial.write(ch);
if (ch == keyword[current_char])
if (++current_char == keyword_length)
{
Serial.println();
return true;
}
}
}
return false;  // Timed out
}
// Read and echo all available module output.
// (Used when we're indifferent to "OK" vs. "no change" responses or to get around firmware bugs.)
void echoFlush()
{while(myWifi.available()) Serial.write(myWifi.read());}
// Echo module output until 3 newlines encountered.
// (Used when we're indifferent to "OK" vs. "no change" responses.)
void echoSkip()
{
echoFind("\n");       // Search for nl at end of command echo
echoFind("\n");       // Search for 2nd nl at end of response.
echoFind("\n");       // Search for 3rd nl at end of blank line.
```

```
}
                // Send a command to the module and wait for acknowledgement
string
                // (or flush module output if no ack specified).
                // Echoes all data received to the serial monitor.
                boolean echoCommand(String cmd, String ack, boolean halt_on_fail)
                {
                myWifi.println(cmd);
                #ifdef ECHO_COMMANDS
                Serial.print("--"); Serial.println(cmd);
                #endif
                // If no ack response specified, skip all available module output.
                 if (ack == "")
                echoSkip();
                else
                // Otherwise wait for ack.
                if (!echoFind(ack))          // timed out waiting for ack string
                if (halt_on_fail)
                errorHalt(cmd+" failed");// Critical failure halt.
                else
                return false;            // Let the caller handle it.
                return true;                 // ack blank or ack found
                }
                // Connect to the specified wireless network.
                boolean connectWiFi()
                {
                String cmd = "AT+CWJAP=" + String("\"") + ssid + String("\"") +
String(",")+String("\"") + pass +String("\"");
                if (echoCommand(cmd, "OK", CONTINUE)) // Join Access Point
                {
                Serial.println("Connected to WiFi.");
                return true;
                }
                else
                {
                Serial.println("Connection to WiFi failed.");
```

```
return false;
}
}
boolean cipmux0()
{
myWifi.println("AT+CIPMUX=0");
Serial.print("Multipleconnect= ");
if (myWifi.find("OK"))
{
Serial.print(" Single mode");
return true;
}
else
{
return false;
}
}
//-------------------------------------------------------------------
boolean cipmode0()
{
myWifi.println("AT+CIPMODE=0");
if (myWifi.find("OK"))
{
return true;
}
else
{
return false;
}
}
//-------------------------------------------------------------------
// ******** SETUP ********
void setup()
{
pinMode(Status, OUTPUT);
pinMode(Buffer, OUTPUT);
```

```
pinMode(pinEN, OUTPUT);
pinMode(rxPin, INPUT);
pinMode(txPin, OUTPUT);
digitalWrite(pinEN, HIGH);
digitalWrite(Status, HIGH);   // turn the LED Status on
 digitalWrite(Buffer, HIGH);   // turn the LED Buffer on
Serial.begin(9600);          // Communication with PC monitor via USB
myWifi.begin(9600);         // Communication with ESP8266 via 5V/3.3V
level shifter
myWifi.setTimeout(TIMEOUT);
Serial.println("\r < Electric energy measuring devices With the online
system >");
 delay(2000);
echoCommand("AT+RST", "OK", CONTINUE);    // Reset & test if the
module is ready
Serial.println("Module is ready.");
echoCommand("AT+GMR", "OK", CONTINUE);   // Retrieves the firmware
ID (version number) of the module.
echoCommand("AT+CWMODE?","OK", CONTINUE);// Get module access
mode.
 echoCommand("AT+CWMODE=1", "", HALT);    // Station mode
echoCommand("AT+CWQAP", "OK", CONTINUE);       // Quit Access
Point
//connect to the wifi
boolean connection_established = false;
for(int i=0;i<5;i++)
{
if(connectWiFi())
{
connection_established = true;
break;
}
}
if (!connection_established) errorHalt("Connection failed");
delay(5000);
```

```
echoCommand("AT+CIFSR", "", HALT);        // Echo IP address.
(Firmware bug - should return "OK".)
if(!cipmux0()) errorHalt("cipmux0 failed");
delay(250);
if(!cipmode0()) errorHalt("cipmode0 failed");
delay(250);
digitalWrite(Buffer, LOW);  //turn the LED Buffer off when set status
module wifi finish.
// current
//emon11.current(1, 60.6);          // Current: input pin, calibration. 33
Ohms
//emon12.current(2, 60.6);          // Current: input pin, calibration. 33
Ohms
emon13.current(3, 60.6);          // Current: input pin, calibration. 33
Ohms
}
// ******** LOOP ********
void loop()
{
delay(100); //just here to slow down the output so it is easier to read
//double Irms1 = emon11.calcIrms(1480);  // Calculate Irms only
//double Irms2 = emon12.calcIrms(1480);  // Calculate Irms only
double Irms3 = emon13.calcIrms(1480);  // Calculate Irms only
// Establish TCP connection
st:
String cmd = "AT+CIPSTART=\"TCP\",\"";  //make this command:
                        AT+CPISTART="TCP","146.227.57.195",80
cmd += serverip;
cmd += "\",80";
myWifi.println(cmd);  //send command to device
delay(2000);  //wait a little while for 'Linked' response - this makes a
difference
if(myWifi.find("Linked"))  //message returned when connection
established WEAK SPOT!! DOESN'T ALWAYS CONNECT
{
Serial.print("Connected to server at ");  //debug message
```

```
Serial.println(serverip);
}
else
{
Serial.println("'Linked' response not received");  //weak spot! Need to
recover elegantly
goto st;
}
// Build HTTP request.
cmd = "GET "+path;
cmd +="/?id="+String(ID);
cmd += "&v1="+String(Irms3)+" HTTP/1.0\r\n\r\n";
delay(2500);
myWifi.print("AT+CIPSEND=");
myWifi.println(cmd.length());  //esp8266 needs to know message
length of incoming message - .length provides this
delay(1500);
if(myWifi.find(">"))    //prompt offered by esp8266
{
Serial.println("sent ok");  //a debug message
myWifi.println(cmd);  //this is our http GET request
Serial.println(cmd);
digitalWrite(Buffer, HIGH); delay(600);
digitalWrite(Buffer, LOW);
}
else
{
digitalWrite(Buffer, HIGH); delay(1000);
myWifi.println("AT+CIPCLOSE");  //doesn't seem to work here?
Serial.println("No '>' prompt received after AT+CPISEND");
digitalWrite(Buffer, LOW); delay(200);
}
myWifi.println("AT+CIPCLOSE");
if(myWifi.find("Unlink"))  //rarely seems to find Unlink? :(
{
Serial.println("Connection Closed Ok...");
```

```
}
else
{
Serial.println("Connection Closed Error !!!");
}
}
```

ข.2 เปลี่ยน ssid , Password , serverip ,path เป็นของเครื่องที่ใช้งาน

```
String ssid = "Techcom08";  // ssid
String pass = "techcom1995"; // Password
//String serverip = "siamcyber.dyndns.info";
String serverip = "172.16.9.179";  // serverip
String path = "/googleChartAPI/lineGrapAPI/apit.php"; // path
```