บทที่ 4
ผลการวิจัย

เนื้อหาในบทนี้จะเกี่ยวกับผลการทดลองในส่วนของการอิมพลิเม้นต์และผลจากการเก็บข้อมูล โดยจะมีรายละเอียดเกี่ยวกับการสร้างแผงวงจร, ชุดคำสั่งในการพัฒนาโปรแกรมส่วนต่าง ๆ และผลการทดลองในแปลงทดลอง สำหรับส่วนหลังของบทนี้จะเป็นการอภิปรายผลการทดลอง เช่น การใช้พลังงาน, ระยะทางการส่งข้อมูล, และ การดึงข้อมูลจากผู้ให้บริการข้อมูลพยากรณ์อากาศ เป็นต้น ดังรายละเอียดต่อไปนี้

4.1 ฮาร์ดแวร์

ผู้วิจัยนำข้อมูลการออกแบบวงจรต่าง ๆ ที่ได้แสดงไว้ในบทที่ 3 มาสร้างเป็นแผงวงจรอิเล็กทรอนิกส์ โดยใช้โปรแกรมดีไซน์สปาร์คพีซีบี (DesignSpark PCB[1]) จากอาร์เอสคอมโพเนนต์ (RS Component) ซึ่งเป็นโปรแกรมสำหรับการออกแบบวงจรที่ให้ผู้ใช้ใช้งานได้ฟรี ผู้วิจัยแปลงสคีมเมติกไดอะแกรมในบทที่ 3 ให้เป็นแผงวงจรจริงโดยเทคนิคดรายฟิล์มซึ่งมีขั้นตอนการทำงานทั่วไปดังนี้

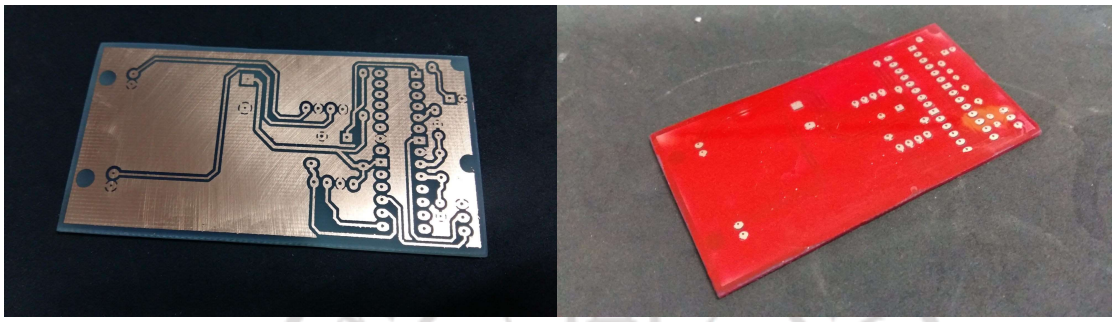
4.1.1 การสร้างแผ่นวงจร

หลังจากแปลงสคีมเมติกไดอะแกรมเป็นลายวงจร, วางตำแหน่งอุปกรณ์ต่าง ๆ และสร้างลายทองแดงเชื่อมต่ออุปกรณ์เรียบร้อยแล้ว จึงสั่งพิมพ์ลายวงจรออกทางเครื่องพิมพ์เลเซอร์ โดยในส่วนของลายทองแดงนั้นสั่งพิมพ์แบบเนกาทิฟ เสร็จแล้วจึงพิมพ์ลายวงจรลงบนแผ่นฟิล์มใส จากนั้นจึงตัดแผ่นวงจรพิมพ์ให้ได้ขนาด ขัดสารเคลือบออกด้วยกระดาษทรายละเอียด ล้างน้ำให้สะอาดแล้วนำไปเคลือบผิวทองแดงด้วยแผ่นดรายฟิล์ม (dry-film) จากนั้นรีดด้วยเครื่องทำบัตรเพื่อให้เรียบและติดกับแผ่นทองแดง เมื่อติดดรายฟิล์มแล้วจึงนำแผ่นใสที่มีลายวงจรมาวางทาบให้ได้ตำแหน่งแล้วนำไปอบในตู้อบแสงยูวีซึ่งทำจากหลอดไฟดักแมลงหรือหลอดไฟไล่ยุงที่สามารถหาซื้อได้ตามร้านค้าปลีกจำหน่ายอุปกรณ์ก่อสร้างทั่วไป จำนวน 4 หลอด ติดตั้งในกล่องซึ่งทำจากกระดาษลูกฟูก (corrugated sheet)

ขั้นตอนต่อไปคือการนำแผ่นวงจรที่ติดดรายฟิล์มไปอบด้วยแสงยูวี (ultraviolet : UV) เพื่อทำให้ดรายฟิล์มส่วนที่โดนแสงยูวีเกิดการเซ็ตตัว ทำการอบเป็นเวลาประมาณ 5 นาที (หรืออาจจะช้าหรือเร็วกว่านี้ ขึ้นอยู่กับคุณสมบัติของตู้อบยูวี) เสร็จแล้วนำมาลอกฟิล์มป้องกันผิวด้านบนดรายฟิล์มออก แล้วจึงนำแผ่นวงจรที่ติดดรายฟิล์มไปแช่ในสารละลายโซดาแอชหรือโซเดียมคาร์บอเนต (sodium carbonate) หลังจากแช่น้ำยาสักครู่นึ่งดรายฟิล์มส่วนที่ไม่โดนแสงยูวีนั้นจะไม่เซ็ตตัวและจะหลุดร่อนออกไป เหลือเฉพาะส่วนที่ต้องการไว้ เมื่อดรายฟิล์มส่วนที่ไม่ต้องการหลุดออกไปหมดแล้วจึงนำแผ่นวงจรไปล้างน้ำ แล้วนำไปแช่ในน้ำยากัดแผ่นวงจรพิมพ์ซึ่งในที่นี้ใช้สารละลายเฟอร์ริกคลอไรด์ (ferric chloride) โดยใช้เวลาแช่ประมาณ 20 - 30 นาที (ขึ้นอยู่กับคุณภาพของสารละลายเฟอร์ริกคลอไรด์และอุณหภูมิแวดล้อม) เสร็จแล้วนำไปล้างในน้ำสะอาด แล้วนำไปแช่ในสารละลายโซดาไฟหรือโซเดียมไฮดรอกไซด์ (sodium hydroxide) เพื่อลอกแผ่นดรายฟิล์มออกจากแผ่นวงจรพิมพ์

1 https://www.rs-online.com/designspark/pcb-software

เสร็จแล้วล้างด้วยน้ำสะอาด ภาพที่ 4.1 (ก) แสดงแผ่นวงจรที่ผ่านการกัดสารละลายเฟอร์ริกคลอไรด์ด้วยแล้ว



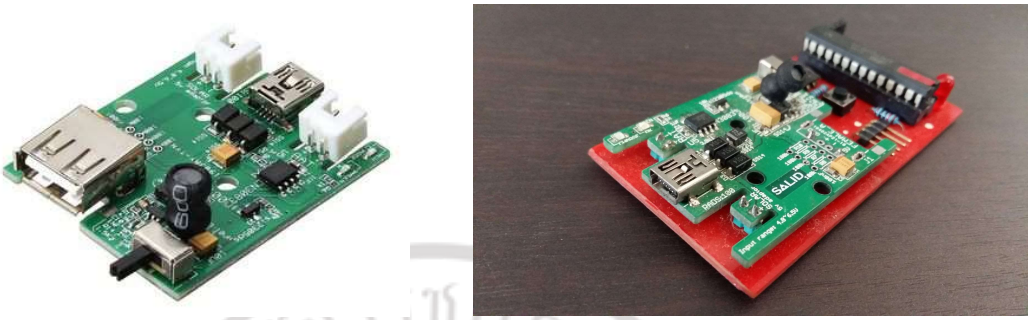
(ก) การกัดลายวงจร (ข) เคลือบโซลเดอร์มาส์กแล้ว

ภาพที่ 4.1 การผลิตแผ่นวงจรโหนดรับบริการ

ขั้นตอนต่อไปคือการเคลือบลายวงจรด้วยโซลเดอร์มาส์ก (solder mask) เพื่อปกป้องทองแดงจากการกัดกร่อนจากสภาพแวดล้อมและเพื่อช่วยให้การบัดกรีอุปกรณ์ต่าง ๆ ทำได้สะดวกมากขึ้น โดยโซลเดอร์มาส์กนั้นอาจจะทำได้ทั้งจากการใช้หมึกที่จะเซ็ตตัวเมื่อโดนแสงยูวีและการใช้แผ่นดรายฟิล์มโซลเดอร์มาส์ก (dry file sodler mask) ในกรณีของหมึกไวแสงยูวีนั้นจะทำด้วยการนำหมึกไวแสงยูวีมาทาลงบนแผ่นวงจรจากนั้นนำแผ่นใสที่มีลายที่ต้องการมาวางทับแล้วนำไปอบในตู้ยูวีเป็นเวลาประมาณ 15 นาที แล้วนำมาเช็ดหมึกส่วนที่ไม่ต้องการด้วยทินเนอร์หรือน้ำมันเบ็นซิน สำหรับกรณีของดรายฟิล์มโซลเดอร์มาส์กนั้นมีขั้นตอนคล้ายกับขั้นตอนการเตรียมลายทองแดง สำหรับงานวิจัยนี้ใช้ทำโซลเดอร์มาส์กโดยการใช้ดรายฟิล์มโซลเดอร์มาส์ก และเมื่อดำเนินการเคลือบโซลเดอร์มาส์กแล้วจึงนำไปเจาะรูเพื่อวางอุปกรณ์ดังภาพที่ 4.1 (ข) จากนั้นติดตั้งอุปกรณ์และบัดกรีให้เรียบร้อยจะได้บอร์ดโหนดรับบริการที่เสร็จสมบูรณ์

4.1.2 บอร์ด

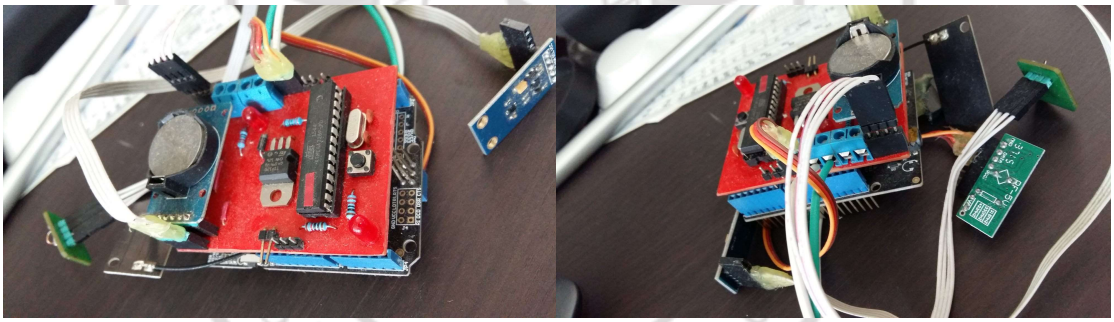
ภาพที่ 4.Error: Reference source not found (ก) ซึ่งแสดงบอร์ดที่ติดตั้งอุปกรณ์ต่าง ๆ เรียบร้อยแล้ว โดยภาพนี้แสดงบอร์ดของโหนดรับบริการ โดยแหล่งพลังงานของโหนดรับบริการนั้นจะใช้แบตเตอรี ขนาดแรงดัน 3.7 โวลต์ และกระแส 1000 มิลลิแอมป์ชั่วโมง ต่อเข้ากับเครื่องประจุแบตเตอรี (battery charger) ซึ่งเชื่อมต่อกับเซลล์สุริยะ (solar cell) ขนาด 3 วัตต์ โดยเครื่องประจุแบตเตอรีนี้จะทำหน้าที่เป็นทั้งเครื่องประจุแบตเตอรีและเป็นเรกูเรเตอร์ปรับแรงดันไฟฟ้าให้เป็น 5 โวลต์ เพื่อส่งให้วงจร โดยในส่วนของเครื่องประจุแบตเตอรีลิโพไรเดอร์นั้นมีการดัดแปลงอุปกรณ์เล็กน้อยเนื่องจากมีอุปกรณ์ต่อพวงและพอร์ตบางส่วนที่ไม่จำเป็น โดยผู้วิจัยนำพอร์ตยูเอสบีและตัวเชื่อมต่อ (connector) ออกเพื่อให้สามารถนำเครื่องประจุแบตเตอรีลิโพไรเดอร์ไปบัดกรีลงบนแผงวงจรที่สร้างได้โดยตรง เพื่อลดขนาดพื้นที่ของอุปกรณ์ ภาพที่ 4.2 แสดงตัวอย่างภาพของเครื่องประจุแบตเตอรีลิโพไรเดอร์ก่อนและหลังการดัดแปลงดังกล่าว



(ก) ก่อนการดัดแปลง (ข) หลังการดัดแปลง

ภาพที่ 4.2 เครื่องประจุแบตเตอรีลิโพไรเดอร์

ภาพที่ 4.3 (ก) แสดงโหนดให้บริการที่เสร็จสมบูรณ์แล้ว และเนื่องจากงานวิจัยนี้ออกแบบให้โหนดให้บริการทำงานตลอดเวลาและมีการติดตั้งอุปกรณ์นี้อยู่ภายในอาคาร ดังนั้นผู้วิจัยจึงออกแบบให้โหนดให้บริการใช้แหล่งพลังงานที่มีแรงดันขนาด 5 โวลต์ จากแหล่งจ่ายไฟภายนอกโดยตรง เช่น อะแดปเตอร์แปลงไฟกระแสสลับเป็นกระแสตรง (AC/DC adaptor) หรือพาวเวอร์แบงค์ (powerbank) ที่ต่อพ่วงมาจากภายในอาคารแล้วต่อเข้ากับโหนดบริการผ่านด้วยเทอร์มินอลบล็อกแบบสกรู (screw terminal block) ขนาดระยะห่าง 5 มิลลิเมตร ดังแสดงในภาพที่ 4.3 (ข)



(ก) (ข)

ภาพที่ 4.3 โหนดให้บริการ

4.1.3 กล่องต้นแบบ

ผู้วิจัยนำวงจรที่ผลิตได้ใส่ในกล่องต้นแบบ โดยได้มีการออกแบบกล่องบรรจุด้วยโปรแกรมออกแบบสามมิติดีไซน์สปาร์คเมคคานิคคัล (DesignSpark PCB[2]) จากอาร์เอสคอมโพเนนต์ แล้วนำแบบจำลองสามมิติที่ออกแบบมาสร้างเป็นวัตถุต้นแบบสามมิติด้วยการใช้เครื่องพิมพ์สามมิติ (3D printer) แบบที่ใช้หลักการเอฟเอฟเอฟ (fused filament fabrication : FFF) หรือเอฟดีเอ็ม (fused deposition modeling : FDM) ซึ่งเป็นสถาปัตยกรรมแบบโอเพนซอร์สฮาร์ดแวร์ (opensource hardware) มีราคาในช่วง 10,000 บาท และมีต้นทุนในการผลิตชิ้นส่วนต่ำ เหมาะสำหรับการนำมาสร้างต้นแบบในงานนี้ การสร้างวัตถุสามมิติทำได้ด้วยการส่งออกไฟล์แบบจำลองใน

2 https://www.rs-online.com/designspark/mechanical-software

รูปแบบเอสทีแอล (stereolithography : STL) จากนั้นจึงนำเข้าไปในโปรแกรมสไลเซอร์[3] (Slic3r) เวอร์ชั่น 1.2.9 ซึ่งเป็นโปรแกรมประยุกต์แบบโอเพนซอร์ส (opensource) เพื่อใช้ในการสไลซ์ (slice) แบบจำลองให้เป็นชั้น โดยกำหนดพารามิเตอร์ที่จำเป็นคืน ความเร็ว จำนวนชั้น ความสูงชั้น หลังจากนั้นจึงส่งออกแบบจำลองที่สไลซ์แล้วเป็นไฟล์จีโค๊ด (G-code) ซึ่งสามารถนำไปสั่งพิมพ์สามมิติได้ การพิมพ์ชิ้นส่วนสามมิติทำในโปรแกรมประยุกต์พรินต์รัน[4] (Printrun) เวอร์ชั่น 1.6.0 ซึ่งเป็นโปรแกรกมประยุกต์แบบโอเพนซอร์สใช้สำหรับการติดต่อและส่งคำสั่งจีโค๊ดไปยังเครื่องพิมพ์สามมิติที่ใช้เฟิร์มแวร์ (firmware) แบบโอเพนซอร์ส เช่น พรูซา (Prusa) และเมอร์ลิน (Merlin) เป็นต้น

ภาพที่ 4.4 (ก) แสดงแบบจำลองสามมิติกล่องใส่อุปกรณ์ต้นแบบที่พิมพ์ได้จากเครื่องพิมพ์สามมิติ ภาพที่ 4.4 (ข) แสดงวัตถุที่พิมพ์ได้จากเครื่องพิมพ์โดยมีการติดตั้งตัวจับ (ชิ้นสีขาว) บริเวณด้านบนเพื่อใช้ยึดติดกับเสาที่ทำจากท่ออะลูมิเนียมขนาดเส้นผ่านศูนย์กลาง 1/2 นิ้ว และภาพที่ 4.4 (ค) แสดงอีกตัวอย่างหนึ่งซึ่งปิดฝาด้านบนเพื่อใช้ในกรณีที่ไม่ต้องการเสา เช่น การนำไปปักลงบนดินโดยตรง

(ก) แบบจำลองสามมิติ (ข) แบบมีตัวจับด้านบน (ค) แบบไม่มีตัวจับ

ภาพที่ 4.4 กล่องต้นแบบสำหรับใส่เซ็นเซอร์วัดข้อมูลดิน

4.2 การอิมพลิเม้นต์ซอฟต์แวร์

เนื้อหาในหัวข้อนี้จะเกี่ยวกับการอิมพลิเม้นต์ชุดคำสั่งต่าง ๆ ที่ได้ออกแบบไว้ในบทที่ 3 ซึ่งจะประกอบด้วยชุดคำสั่งบนโหนดรับบริการ, โหนดให้บริการ และบนคอมพิวเตอร์ให้บริการ มีการใช้งานคลังโปรแกรมภายนอก (third-party library) จากแหล่งต่าง ๆ เพื่อให้โปรแกรมทำงานได้ตาม

3 http://slic3r.org/

4 http://www.pronterface.com/

ต้องการหลายชุด ดังแสดงในตารางที่ 4.1 และมีรายละเอียดในส่วนต่าง ๆ ที่สำคัญและควรกล่าวถึงดังต่อไปนี้

ตารางที่ 4.1 คลังโปรแกรมภายนอกที่ใช้ในงานวิจัยนี้

คลังโปรแกรม	รายละเอียดส่วนที่มีการใช้งาน	ผู้พัฒนา
GPRS_Shield_Arduino	มอดูลจีพีอาร์แอส/จีเอสเอ็ม	Lawliet Zou (2014)
VirtualWire	ตัวรับ/ส่งวิทยุ	Mike McCauley (2013)
Wire	มอดูลนาฬิกาจริง	Arduino
SoftwareSerial	การอ่านข้อมูลแบบอนุกรม	Mike Hart
NewSoftSerial library		
SHT1x	มอดูลเซ็นเซอร์อากาศ	Jonathan Oxer (2009) และ Maurice Ribble (2008)
Suli	คลังโปรแกรมสำหรับมอดูลจาก Seeed Studio	Seeed Studio
cURL	อ่านข้อมูลพยากรณ์อากาศ	Daniel Stenberg (2018)

4.2.1 ชุดคำสั่งที่สำคัญบนโหนดรับบริการ

1) การเข้ารหัสข้อมูล

เนื่องจากฟังก์ชันสำหรับแปลงชนิดข้อมูลที่มีอยู่ในคลังโปรแกรมมาตรฐานของ Arduino มีข้อจำกัดหลายอย่าง ข้อมูลที่ได้บางส่วนไม่อาจจะนำไปใช้ได้โดยตรงและบางส่วนไม่ตรงตามความต้องการของงานวิจัยนี้

```
1    /**
2     * Encode 4-byte float to 2-byte array of byte
3     */
4    void encode_data(float *val, byte *encoded_data) {
5      int tmp;
6      tmp = int((*val) * 100);
7      *((int *)byte_array) = tmp;
8      encoded_data[0] = byte_array[0];
9      encoded_data[1] = byte_array[1];
10   }
```

ภาพที่ 4.5 ฟังก์ชัน encode_data

ดังนี้ผู้วิจัยจึงออกแบบฟังก์ชัน encode_data เพื่อทำหน้าที่เข้ารหัสหรือแปลงข้อมูลจากตัวแปรชนิด float ที่อยู่ในช่วง [0.00, 99.99] ให้เป็นแถวลำดับของตัวแปร byte

ขนาด 2 ไบต์ โดยฟังก์ชันนี้มีพารามิเตอร์ 2 ค่า ประกอบด้วย `val` ซึ่งเป็นตัวชี้ไปยังตัวแปรแบบ `float` ซึ่งเก็บค่าที่ต้องการนำมาเข้ารหัส และค่าที่สองคือ `encoded_data` เป็นตัวชี้ของตัวแปรแบบ `byte` สำหรับบันทึกผลลัพธ์ของการเข้ารหัส โดยการแปลงชนิดข้อมูลระหว่าง `float` และ `byte` นั้นจะทำด้วยการคูณค่าที่ให้มาด้วย 100 เพื่อเลื่อนทศนิยม 2 หลักแรกให้กลายมาเป็นจำนวนเต็มหลักหน่วยและหลักสิบ ตัวอย่างเช่น แปลงจากค่า 25.61 ให้กลายเป็น 2561 จากนั้นจึงคัดลอกค่าที่แปลงได้ไปเก็บไว้ยังบัฟเฟอร์สำรองข้อมูลชั่วครว `byte_array` โดยใช้เทคนิคการคัดลอกผ่านตัวชี้ เสร็จแล้วจึงคัดลอกเฉพาะสองไบต์แรกในบัฟเฟอร์ชั่วคราวซึ่งเป็นข้อมูลส่วนของเลขจำนวนเต็มไปเก็บในตัวแปรที่ส่งมา ดังภาพที่ 4.5

2) รอบการทำงานหลัก

ฟังก์ชัน `loop` ของโหนดรับบริการจะทำงานตลอดเวลา โดยเมื่อมีการเรียกใช้งานจะเริ่มต้นด้วยการอ่านข้อมูลอุณหภูมิและความชื้นสัมพัทธ์ของดินจากเซ็นเซอร์ จากนั้นจึงเข้ารหัสข้อมูลให้เป็นแถวลำดับของไบต์แล้วนำมาเก็บในบัฟเฟอร์สำหรับเตรียมส่งไปให้โหนดให้บริการ โดยข้อมูลนี้จะบันทึกจะมีโครงสร้างการจัดเก็บตามรายละเอียดที่กล่าวมาแล้ว โดยในกรณีค่าอุณหภูมิและความชื้นสัมพัทธ์ของดินจะบันทึกไว้ในตัวแปร `msg` ที่ไบต์ที่ 5, 6, 7 และ 8 ตามลำดับ หลังจากนั้นจึงใช้ฟังก์ชัน `analogRead` เพื่ออ่านค่าระดับแรงดันไฟฟ้าปัจจุบันของบอร์ดมาแปลงและบันทึกไว้ในตัวแปร `msg` ลำดับที่ 9 และ 10 ตามลำดับ และข้อมูลส่วนสุดท้ายคือลำดับหรือตัวนับของการส่งข้อมูลของโหนดรับบริการ ซึ่งจะบันทึกในตัวแปร `msg` ลำดับที่ 11 โดยค่านี้เป็นเลขจำนวนเต็มบวกที่จะเพิ่มค่าทุกครั้งที่มีการส่งข้อมูล ใช้เพื่อตรวจสอบว่ามีการส่งข้อมูลครบถ้วนทุกครั้งหรือไม่ โดยจะมีการรีเซตค่าเป็น 0 เมื่อมีการส่งข้อมูลครบ 10 ครั้ง

เมื่อเตรียมตัวแปร `msg` เรียบร้อยแล้วจึงส่งข้อมูลนี้ให้ตัวส่งวิทยุ ด้วยการส่งตัวแปรนี้ไปให้ฟังก์ชัน `vw_send` พร้อมกับข้อมูลขนาดหรือความยาวของข้อความนี้ หลังจากนั้นจึงเรียกใช้ฟังก์ชัน `vw_wait` เพื่อรอให้ตัวส่งวิทยุส่งข้อมูลให้เรียบร้อย โดยระหว่างการส่งข้อมูลนี้จะมีการส่งสัญญาณให้แอลอีดีของตัวส่งวิทยุทำงานและปิดเมื่อไมโครคอนโทรเลอร์ส่งข้อมูลเสร็จ อย่างไรก็ตามผู้วิจัยพบว่าบางครั้งการส่งข้อมูลทำได้ไม่สมบูรณ์ ข้อมูลขาดหาย ดังนั้นจึงกำหนดให้โปรแกรมวนซ้ำเพื่อส่งค่าไปยังโหนดให้บริการให้มากกว่า 1 ครั้ง โดยกำหนดจำนวนครั้งไว้ในค่าคงที่ `RF_SEND_REPEAT_TIME` ซึ่งงานวิจัยนี้กำหนดให้มีค่าเป็น 5 ครั้ง และมีการหยุดรอระหว่างการส่งแต่ละครั้งเป็นเวลา 1000 มิลลิวินาที เมื่อผ่านขั้นตอนการส่งข้อมูลแล้วจึงเข้าสู่โหมดการประหยัดพลังงานตามระยะเวลาที่กำหนดก่อนที่จะตื่นมาเพื่อทำงานรอบต่อไป ภาพที่ 4.6 การทำงานภายในฟังก์ชัน `loop`


```
1    /**
2     * Main loop
3     */
4    void loop()
5    {
6      int i;
7      int val;
8
9      // Measure soil data : temperature and humidity
10     ...
```

```
11
12      // Soil RH
13      msg[5] = byte_array_rh[0];
14      msg[6] = byte_array_rh[1];
15
16      // Soil temperature.
17      msg[7] = byte_array_c[0];
18      msg[8] = byte_array_c[1];
19
20      // Battert voltage
21      val = analogRead(BATTERY_PIN);
22      float t = val/100.0;
23      encode_data(&t, &byte_array_rh[0]);
24      msg[ 9] = byte_array_rh[0];
25      msg[10] = byte_array_rh[1];
26
27      // Counter.
28      t = count/100.0;
29      encode_data(&t, &byte_array_rh[0]);
30      msg[11] = byte_array_rh[0];
31
32      // Try to send several times.
33      for(i = 0; i < RF_SEND_REPEAT_TIME; i++) {
34        digitalWrite(LED_PIN_RF, HIGH);
35        vw_send((uint8_t *)msg, MSG_LENGTH);
36        vw_wait_tx();
37        digitalWrite(LED_PIN_RF, LOW);
38        delay(1000);
39      }
40
41      // Go to low power mode.
42      ...
43
44      count = count + 1;
45      if(count > 9) {
46        count = 0;
47      }
48    }
```

ภาพที่ 4.6 ฟังก์ชัน `loop`

3) การประหยัดพลังงาน

ภาพที่ 4.7 แสดงการอิมพลิเม้นต์การประหยัดพลังงานตามหัวข้อที่ 3.3.4 ซึ่งทำด้วยคลังโปรแกรม `LowPower` โดยใช้ฟังก์ชัน `powerDown` มีการส่งอาร์กิวเม้นต์สามค่า ค่าแรกคือระยะเวลาของการหลับ ซึ่งในที่นี้กำหนดด้วยค่าคงที่ `SLEEP_8S` ซึ่งจะทำให้วงจรหยุดการทำงานและหลับเป็นเวลา 8 วินาที สำหรับอาร์กิวเม้นต์ค่าที่สองคือการสั่งหยุดการทำงานของตัวแปลงแอนะล็อกเป็นดิจิทัลด้วยค่าคงที่ `ADC_OFF` และอาร์กิวเม้นต์สุดท้ายคือการยกเลิกการทำงานของตัวตรวจหาระดับแรงดันไฟหรือบราวน์เอ๊าต์ดีเทคเตอร์ (brown out detector) ซึ่งในกรณีนี้กำหนดให้ปิดการทำงานของบราวน์เอ๊าต์ดีเทคเตอร์ด้วยค่าคงที่ `BOD_OFF`

และเนื่องจากเราสามารถสั่งให้ไมโครคอนโทรเลอร์หยุดพักการทำงานได้เต็มที่เป็นเวลาเพียง 8 วินาทีเท่านั้น ดังนั้นการที่จะทำให้ไมโครคอนโทรเลอร์หยุดการทำงานได้เท่ากับระยะเวลาที่กำหนดไว้นานมากกว่านั้นจึงจำเป็นต้องใช้การวนซ้ำเพื่อสั่งให้ไมโครคอนโทรเลอร์หลับตามจำนวนรอบที่กำหนดไว้ งานวิจัยนี้กำหนดค่านี้ในค่าคงที่ NUM_SLEEP_ITERATIONS ซึ่งงานวิจัยนี้ต้องการให้โหนดรับบริการทำการวัดข้อมูลทุก ๆ 10 นาที ดังนั้นผู้วิจัยจึงได้กำหนดค่าคงที่ NUM_SLEEP_ITERATIONS ให้มีค่าเท่ากับ (10*60)/8 หรือ 75 รอบ

```
1    ...
2    // Go to low power mode.
3    for(i = 0; i < NUM_SLEEP_ITERATIONS; ++i) {
4      LowPower.powerDown(
5        SLEEP_8S, // sleep for 8s
6        ADC_OFF, // turn off ADC module
7        BOD_OFF  // turn off Brown Out Detector (BOD) module.
8      );
9    }
10   ...
```

ภาพที่ 4.7 ชุดคำสั่งควบคุมการหลับของไมโครคอนโทรเลอร์

4.2.2 ชุดคำสั่งที่สำคัญบนโหนดให้บริการ

ดังที่ได้กล่าวมาแล้วว่างานวิจัยนี้ใช้ไมโครคอนโทรเลอร์ ATmega328P-PU ซึ่งมีหน่วยความจำเอสแรม (SRAM) สำหรับการใช้งานเพียง 2 กิโลไบต์ (Arduino, 2018) ดังนั้นเพื่อให้การใช้หน่วยความจำมีประสิทธิภาพ จึงออกแบบให้ค่าคงที่สายอักขระต่าง ๆ ด้วยการกำหนดเป็นตัวแปรอักขระความกว้างคงที่ (fixed width) ซึ่งจะช่วยประหยัดหน่วยความจำได้ (Adafruit, 2013)

```
1    ...
2    void loop() {
3      ...
4      // The receiver has recived a message.
5      // Extract the message and some additional data
6      // and send them to the server.
7      if (vw_get_message(buf, &buflen)) {
8        blinker.blink(LED_PIN, 5, LED_PAUSE_INTERVAL_SHORT);
9
10       prepare_data(); // prepare data
11       send_data(); // send data.
12
13       ...
14
15       // Done.
16       blinker.blink(LED_PIN, 5, LED_PAUSE_INTERVAL_SHORT);
17       }
18   }
```

ภาพที่ 4.8 ชุดคำสั่งอ่านข้อมูลและส่งข้อมูล

ภาพที่ 4.8 แสดงชุดคำสั่งของการวนซ้ำหลัก ซึ่งไมโครคอนโทรเลอร์จะมีการเรียกใช้ฟังก์ชัน vw_get_message เพื่อตรวจสอบว่ามีสายอักขระเข้ามาจากตัวรับวิทยุหรือไม่ ถ้ามีไมโครคอนโทรเลอร์จะกระพริบแอลอีดีสถานะ 5 ครั้งแล้วเรียกฟังก์ชัน prepare_data เพื่ออ่านข้อมูลและเตรียมข้อมูล จากนั้นจึงเรียกฟังก์ชัน send_data เพื่อดำเนินการส่งข้อมูลไปยังเครื่องคอมพิวเตอร์ให้บริการ เสร็จแล้วจึงกระพริบแอลอีดีสถานะ 5 ครั้งก่อนที่จะวนกลับไปตรวจสอบสายอักขระเครื่องรับวิทยุรอบใหม่

1) การเตรียมข้อมูล

ฟังก์ชัน prepare_data จะเริ่มต้นอ่านข้อมูลวันที่และเวลาจากอุปกรณ์ด้วยการเรียกเมท็อด get_datetime ของคลาส DS3231 พร้อมกับส่งตัวแปรแถวลำดับ date_time ไปให้บันทึกข้อมูลกลับมา แล้วแปลงข้อมูลให้เป็นข้อมูลแบบบีดีซี ตามแสดงในภาพที่ 4.9 โดยจะใช้ฟังก์ชัน itoa จากคลังโปรแกรมมาตรฐานเพื่อแปลงข้อมูลปี ค.ศ. ในตัวแปร date_time ซึ่งเป็นตัวแปรเลขจำนวนเต็มให้กลายเป็นตัวแปรแถวลำดับของอักขระ tmp_data_uc ในรูปแบบค่าตัวเลขฐานสิบ หลังจากนั้นจึงนำข้อมูลในตัวแปร date_time และ tmp_data_uc ส่งต่อไปให้ฟังก์ชัน fill_date_time_data

ฟังก์ชัน fill_date_time_data จะมีพารามิเตอร์ 3 ค่าในรูปแบบของตัวชี้ทั้งหมด ค่าแรกคือข้อมูลต้นฉบับใช้สำหรับเปรียบเทียบช่วงข้อมูลว่าอยู่หลักหน่วยหรือหลักสิบ ใช้สำหรับการเติมศูนย์ (zero fill) นำหน้าในกรณีที่ข้อมูลที่ส่งมามีค่าน้อยกว่า 10 สำหรับค่าที่สองและสามคือตัวแปรที่พักข้อมูลชั่วคราวที่ต้องการแปลงค่าและตัวแปรเก็บผลลัพธ์ตามลำดับ

```
1    ...
2    void loop() {
3      ..
4      // Year
5      itoa(date_time[0], (char *)&tmp_data_uc[0], 10);
6      fill_date_time_data(&date_time[0],
7                          &tmp_data_uc[0],
8                          (byte *)&out_msg[27]);
9      ..
10   }
11   ...
12   /**
13    * Fill date/time data
14    */
15   void fill_date_time_data(byte *c, byte *s, byte *d) {
16     if((*c) < 10) {
17       d[0] = '0';
18       d[1] = s[0];
19     } else {
20       d[0] = s[0];
21       d[1] = s[1];
22     }
23   }
24   ...
```

ภาพที่ 4.9 ชุดคำสั่งการแปลงและเข้ารหัสข้อมูลแบบสองไบต์

ภาพที่ 4.11 แสดงขั้นตอนในการวนซ้ำหลัก โดยไมโครคอนโทรเลอร์อ่านข้อมูลอุณหภูมิและความชื้นสัมพัทธ์ของอากาศจากคลังโปรแกรม SHT1x ด้วยแนวทางเดียวกัน นั่นคือจะใช้ตัวแปร air_temp และ air_rh เพื่อเก็บค่าอุณหภูมิและความชื้นสัมพัทธ์ของอากาศจากเซ็นเซอร์ แล้วส่งค่าที่ได้ไปแปลงและจัดลำดับด้วยฟังก์ชัน fill_sensor_data_4 โดยเลข 4 ในชื่อฟังก์ชันหมายถึงการแปลงข้อมูลให้เป็นสายอักขระจำนวนสี่ไบต์

```
1    ...
2    void loop() {
3      ..
4      air_temp = sht11.readTemperatureC();
5      air_rh = sht11.readHumidity();
6      fill_sensor_data_4(&out_msg[10], air_rh);
7      fill_sensor_data_4(&out_msg[14], air_temp);
8      ..
9    }
10   ...
```

ภาพที่ 4.10 ชุดคำสั่งการจัดการข้อมูลอุณหภูมิและความชื้นสัมพันธ์อากาส

ภาพที่ 4.11 แสดงการอิมพลิเม้นต์ฟังก์ชัน fill_sensor_data_4 ซึ่งมีพารามิเตอร์สองค่าคือ b ซึ่งเป็นตัวชี้ของบัฟเฟอร์ข้อมูลผลลัพธ์และ val ซึ่งเป็นข้อมูลอินพุจ ฟังก์ชันแปลงข้อมูลในตัวแปร val ที่ส่งเข้ามาจากเลขทศนิยมให้เป็นเลขจำนวนเต็มเก็บไว้ในตัวแปร tmp จากนั้นจึงทำการแปลงค่าจาก tmp ซึ่งเป็นเลขจำนวนเต็มให้เป็นตัวแปรสายอักขระด้วยฟังก์ชัน itoa ในการแปลงข้อมูลแต่ละครั้งจะใช้พื้นที่สี่ไบต์ โดยข้อมูลสองไบต์แรกเป็นค่าของส่วนเลขจำนวนเต็ม โดยจะมีการตรวจสอบว่าค่าเลขจำนวนเต็มนี้น้อยกว่า 10 หรือไม่ หากน้อยกว่าจริงก็จะทำการเติมศูนย์ที่หลักแรก ดังชุดคำสั่งในภาพที่ 4.11 บรรทัดที่ 9 - 15 และงานวิจัยนี้ออกแบบให้บันทึกค่าทศนิยมเพียงสองหลัก ดังนั้นโปรแกรมคำนวณผลต่างและคูณด้วย 100 เพื่อเลื่อนทศนิยมสองหลักแรกให้ขึ้นมาเป็นเลขจำนวนเต็มสองหลักแรก จากนั้นจึงแปลงให้เป็นสายอักขระด้วยแนวทางเดียวกับกรณีแรก เสร็จแล้วบันทึกค่าที่ได้ที่สองไบต์หลังของแถวลำดับที่ส่งมา ดังชุดคำสั่งในภาพที่ 4.11 บรรทัดที่ 16 - 25

```
1    ...
2    void fill_sensor_data_4(char *b, float val) {
3      unsigned int tmp;
4      unsigned char bb[4];
5
6      // Integer part.
7      tmp = (int)val;
8      itoa(tmp, (char *)&bb[0], 10);
9      if(tmp < 10) {
10       b[0] = '0';
11       b[1] = (char)bb[0];
12     } else {
13       b[0] = (char)bb[0];
14       b[1] = (char)bb[1];
```

```
15      }
16      tmp = (int)((val - floor(val))*100.0);
17      itoa(tmp, (char *)&bb[0], 10);
18      if(tmp < 10) {
19        b[2] = '0';
20        b[3] = (char)bb[0];
21      } else {
22        b[2] = (char)bb[0];
23        b[3] = (char)bb[1];
24      }
25    }
```

ภาพที่ 4.11 ชุดคำสั่งการแปลงและเข้ารหัสข้อมูลแบบสี่ไบต์

2) การส่งข้อมูลผ่านจีพีอาร์เอส

การอิมพลิเม้นต์การส่งข้อมูลผ่านจีพีอาร์เอสเริ่มต้นด้วยการกำหนด อัตราบอด (buad rate) สำหรับการสื่อสารระหว่างไมโครคอนโทรเลอร์และมอดูลจีพีอาร์แอส/จีเอสเอ็มไว้ที่ 9600 บิตต่อวินาที ซึ่งเป็นค่ามาตรฐานโดยปริยายสำหรับการสื่อสารของมอดูลที่ใช้ในงานวิจัยนี้ ผู้วิจัยสร้างตัวแปรสายอักขระ `char url[] = "org.rbru.ac.th";` เป็นตัวแปรสำหรับเก็บข้อมูลยูอาร์แอลของเครื่องคอมพิวเตอร์ให้บริการ ซึ่งงานวิจัยนี้กำหนดเป็นเครื่องให้บริการของสาขาวิชาภูมิสารสนเทศที่ภายในศูนย์เทคโนโลยีสารสนเทศ สำนักวิทยบริการและเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏรำไพพรรณี และมีตัวแปรสายอักขระ http_cmd เพื่อเก็บข้อความร้องขอตามที่ได้กล่าวมาแล้วในบทที่ 3

เมื่อเริ่มต้นทำงาน โปรแกรมจะสร้างตัวแปร `gprs` ซึ่งเป็นวัตถุของคลาส `GPRS_Shield_Arduino` (Zou, 2014) โดยในขั้นตอนการจัดเตรียมในฟังก์ชัน `setup` จะมีการสั่งให้อุปกรณ์ทำงานและรอจนกว่ามอดูลจีพีอาร์แอสจะเชื่อมต่อเข้ากับเครือข่ายโทรศัพท์ได้สำเร็จ โดยในระหว่างการรอนั้นจะสั่งให้แอลอีดีสถานะกระพริบอยู่ตลอดเวลา ดังตัวอย่างในภาพที่ 4.12

```
1   void setup() {
2     ...
3     gprs.init();
4     while(false == gprs.join(F("cmnet"))) {
5       blinker.blink(LED_PIN, 5, LED_INTERVAL_LONG);
6     }
7     ...
8   }
```


ภาพที่ 4.12 ชุดคำสั่งควบคุมการหลับของไมโครคอนโทรเลอร์

โดยไมโครคอนโทรเลอร์จะตรวจสอบสถานะของมอดูลจีพีอาร์แอส/จีเอสเอ็มว่าพร้อมทำงานหรือไม่ จากนั้นจะตรวจสอบสถานะการเชื่อมต่อกับเครือข่ายโทรศัพท์ เสร็จแล้วจึงเชื่อมต่อกับบริการด้วยการเปิดการเชื่อมต่อแบบทีซีพี (transmission control protocol : TCP) พร้อมกับส่งยูอาร์แอลเป้าหมายรวมทั้งหมายเลขพอร์ตของการสื่อสารไปให้เมท็อด `connect` หลังจากเชื่อมต่อสำเร็จแล้วจึงคัดลอกข้อมูลจากตัวแปร `out_message` มาใส่ในตัวแปร

http_cmd แล้วส่งข้อมูลในตัวแปร http_cmd ออกไป ด้วยการเรียกใช้เมท็อด send ของคลาส GPRS_Shield_Arduino ดังชุดคำสั่งในภาพที่ 4.13 บรรทัดที่ 6 - 10 โดยจะมีการคำนวณ sizeof(httpd_cmd) - 1 เพื่อนับจำนวนไบต์หรือความยาวของสายอักขระที่จะส่ง จากนั้นจะใช้คำสั่ง while เพื่อรอข้อความตอบรับจากเครื่องคอมพิวเตอร์ให้บริการ เมื่อจัดการข้อความตอบรับจากเครื่องคอมพิวเตอร์ให้บริการแล้วจะปิดการเชื่อมต่อ ดังชุดคำสั่งในภาพที่ 4.13 บรรทัดที่ 12 - 15

```
1    void send_data() {
2      ...
3      gprs.join(F("cmnet")));
4      gprs.connect(TCP, url, 80);
5
6      // Copy data and send
7      for(i = 0; i < OUT_MESSAGE_LEN; i++) {
8        http_cmd[58+i] = (char)out_msg[i];
9      }
10     gprs.send(http_cmd, sizeof(http_cmd)-1);
11
12     // Wait for response from server
13     while (true) {
14       ...
15     }
16
17     // Done
18     gprs.close();
19     gprs.disconnect();
20     ...
21   }
```

ภาพที่ 4.13 ชุดคำสั่งควบคุมการเตรียมข้อมูลและส่งข้อมูลผ่านจีพีอาร์แอส

การตรวจสอบข้อความตอบรับจากเครื่องคอมพิวเตอร์ให้บริการนั้นจะใช้เทคนิคการตรวจสอบข้อความตอบรับ ซึ่งข้อความตอบกลับนี้เป็นไปตามมาตรฐานโพรโทคอลเอชทีทีพี (hypertext transfer protocol : HTTP) นั่นคือจะประกอบด้วยข้อมูลสองส่วนส่งมาเรียงตามลำดับ โดยข้อมูลส่วนแรกที่ส่งกลับมาก่อนคือส่วนหัวตอบรับ (response header) ซึ่งจะมีรายละเอียดเกี่ยวกับข้อมูลที่ส่งตามกลับมาหลังจากนี้ เช่น สถานะและความยาวของข้อความที่ส่งกลับ และข้อมูลส่วนที่สองคือข้อมูลหรือเนื้อหาที่ส่งมาจากเครื่องคอมพิวเตอร์ให้บริการที่ผู้ใช้ต้องการ การตรวจสอบข้อมูลส่วนแรกนั้นทำด้วยการใช้ตัวแปร ret เพื่อตรวจสอบค่าที่คืนมาจากเมท็อด recv โดยโปรแกรมจะวนรอบรอจนกว่าจะได้รับค่าที่กำหนดไว้ ซึ่งในงานวิจัยจะต้องมีค่าที่คืนมาเป็น 127 และ 45 ตามลำดับ

ผู้วิจัยกำหนดค่าทั้งสองนี้เพื่อใช้สำหรับบอกว่าโปรแกรมได้รับข้อมูลส่วนหัวและข้อมูลแล้วหรือยัง หากพบว่าได้รับข้อมูลที่มีความยาวดังกล่าวถูกต้องแล้วแสดงว่าได้รับข้อมูลจากเครื่องคอมพิวเตอร์ให้บริการทั้งในส่วนของข้อมูลส่วนหัวและข้อมูลเรียบร้อย ดังนั้นโปรแกรมจะสกัดข้อมูลดิบในตัวแปร buffer ออกมา โดยข้อมูลที่จำเป็นต่อการใช้งานต่อคือวันที่ปัจจุบัน ซึ่งอยู่ใน

ตัวแปร buffer ลำดับที่ 38 และมีความยาว 3 ไบต์ ข้อมูลที่สองคือระยะเวลาการให้น้ำ ซึ่งอยู่ในตัวแปร buffer ลำดับที่ 41 และมีความยาว 5 ไบต์ โดยโปรแกรมจะแปลงข้อมูลดิบหรือสายอักขระนี้ให้อยู่เป็นเลขจำนวนเต็มด้วยการเรียกใช้ฟังก์ชัน my_text_to_int เสร็จแล้วจึงส่งข้อมูลเหล่านี้ไปให้ฟังก์ชันส่วนที่เกี่ยวข้องกับการควบคุมการเปิด/ปิดวาวล์น้ำทำงานต่อไป ดังรายละเอียดในภาพที่ 4.14

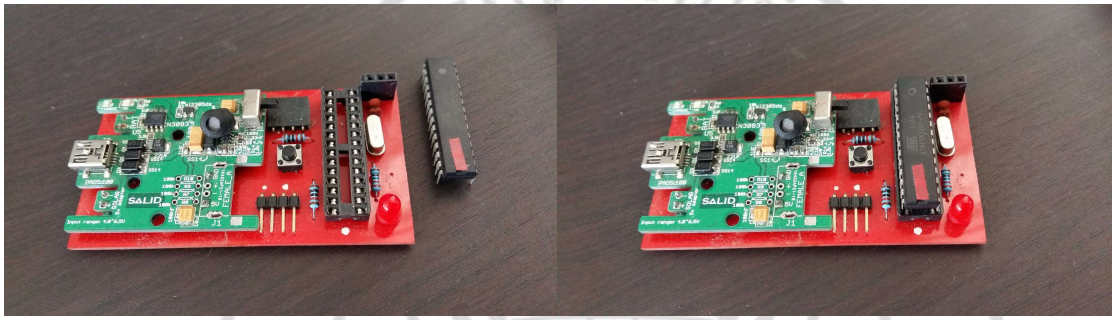
```
1    ...
2    while (true) {
3     int ret = gprs.recv(buffer, sizeof(buffer)-1);
4
5     // 1st section
6     if (ret == 127) { num_returned_byte_1 = 127; }
7
8     // 2nd section
9     if(ret == 45) {
10    if(num_returned_byte_1 == 127) {
11    irr_doy_new = my_text_to_int(&buffer[38], 3);
12    irr_duration = my_text_to_int(&buffer[41], 4);
13
14    // Check if we required to open the valve.
15    if(irr_doy_new != irr_doy_old) {
16          // Open valve
17          digitalWrite(LED_PIN_SOLENOID, HIGH);
18          delay(irr_duration);
19          digitalWrite(LED_PIN_SOLENOID, LOW);
20
21          // Update current doy
22          irr_doy_old = irr_doy_new;
23    }
24
25    // Reset
26    num_returned_byte_1 = 0;
27    }
28   }
29   ...
```

ภาพที่ 4.14 ชุดคำสั่งการตรวจสอบการให้น้ำ

4.2.3 การอัพโหลดโปรแกรมลงไมโครคอนโทรเลอร์

วงจรของโหนดรับบริการและโหนดให้บริการที่ใช้ในงานวิจัยนี้เป็นบอร์ดแบบทำงานลำพัง ไม่มีพอร์ตที่จะใช้สำหรับการสื่อสารกับคอมพิวเตอร์โดยตรง การเขียนโปรแกรมใส่ไมโครคอนโทรเลอร์บนวงจรที่ทำเสร็จแล้วโดยตรงจึงทำไม่ได้ เทคนิคที่อาจจะใช้เพื่อการเขียนโปรแกรมคือการเพิ่มช่องทางการเชื่อมต่อโดยต่อลายวงจรเพิ่มอีกสองจุดเพื่อเชื่อมกับขา TX และ RX ของไมโครคอนโทรเลอร์ อย่างไรก็ตามงานวิจัยนี้ใช้ไมโครคอนโทรเลอร์ ATmega328P-PU ซึ่งเป็นไมโครคอนโทรเลอร์ที่ใช้อยู่กับบอร์ดไมโครคอนโทรเลอร์ Arduino UNO อยู่แล้ว และมีรูปแบบขาแบบ DIP-28 และมีบูตโหลดเดอร์ติดตั้งมาเรียบร้อยแล้ว สามารถถอดสลับชิบของบอร์ดที่พัฒนากับชิปบนบอร์ดไมโคร

คอนโทรเลอร์ Arduino UNO ได้โดยตรง ดังนั้นขั้นตอนการอัพโหลดชุดคำสั่งใส่ไมโครคอนโทรเลอร์นั้น ผู้วิจัยจึงใช้วิธีการถอดสลับกับชิบไมโครคอนโทรเลอร์ ATmega328P-PU จากบอร์ดที่พัฒนาภาพที่ 4.15 (ก) แล้วนำชิปไมโครคอนโทรเลอร์ ATmega328P-PU ที่ถอดออกมานี้ไปสลับกับชิปบนบอร์ดไมโครคอนโทรเลอร์ Arduino UNO จากนั้นจึงอัพโหลดโปรแกรมที่ต้องการใส่ในชิปนี้ เสร็จแล้วจึงถอดออกแล้วนำไมโครคอนโทรเลอร์กลับไปติดตั้งในบอร์ดที่พัฒนาตามเดิม ดังแสดงตัวอย่างในภาพที่ 4.15 (ข)



(ก) ถอดชิบ ATmega328P-PU (ข) ใส่กลับ

ภาพที่ 4.15 การเขียนโปรแกรมลงบนไมโครคอนโทรเลอร์

4.2.4 ชุดคำสั่งที่สำคัญบนคอมพิวเตอร์ให้บริการ

ผู้วิจัยพัฒนาชุดคำสั่งบนเครื่องคอมพิวเตอร์ให้บริการซึ่งเป็นเครื่องบริการเว็บ (web server) พัฒนาด้วยภาษาพีเอชพี (PHP: Hypertext Preprocessor : PHP) เวอร์ชัน 5 เป็นหลักการจัดเก็บข้อมูลต่าง ๆ เป็นฐานข้อมูลทำด้วยใช้ระบบจัดการฐานข้อมูลมายเอสคิวแอล (MySQL) และในบางส่วนจะมีการใช้ภาษาจาวาสคริปต์ (JavaScript) เป็นตัวเสริมในการจัดการข้อมูลเพื่อการแสดงผล โดยผู้วิจัยพัฒนาชุดคำสั่งต่าง ๆ โดยไม่ได้ใช้เฟรมเวิร์ค (framework) อื่นเพื่อให้โปรแกรมที่พัฒนาได้นี้กระชับและลดการใช้เฟรมเวิร์คที่อาจจะทำให้โปรแกรมที่พัฒนาได้มีขนาดใหญ่เกินจำเป็น

โดยชุดโปรแกรมที่พัฒนาในส่วนนี้ทั้งหมดติดตั้งและทดสอบผ่านโปรแกรมบริการเว็บอาปาเช่ (Apache) เวอร์ชัน 2.0 ซึ่งในระหว่างการทดสอบการทำงานได้ติดตั้งลงบนระบบปฏิบัติการไมโครซอฟต์วินโดว์ (Microsoft Windows) เวอร์ชั่น 10 และในการทดสอบการทำงานจริงได้ติดตั้งในเครื่องให้บริการของมหาวิทยาลัยซึ่งมีระบบปฏิบัติการอูบุนตู (Ubuntu) เวอร์ชั่น 16.04 การทดสอบการแสดงผลทำด้วยการใช้เว็บเบราเซอร์ไฟร์ฟอกซ์ (Firefox) โดยรายละเอียดเกี่ยวกับการอิมพลิเม้นต์ส่วนที่สำคัญมีดังต่อไปนี้

1) การสกัดข้อมูลพยากรณ์อากาศ

ในงานวิจัยนี้ผู้วิจัยได้ออกแบบให้ระบบดึงข้อมูลพยากรณ์อากาศจากแหล่งข้อมูลต่าง ๆ ให้ผู้ใช้เลือกใช้ตามความต้องการจำนวนทั้งหมดสามแหล่ง ประกอบด้วยกรมอุตุนิยมวิทยา, ฟอร์คา (Foreca[5]) และโอเพ่นเวเธอร์แมพ (OpenWeatherMap[6]) โดยแหล่งข้อมูล

5 https://www.foreca.com

6 https://openweathermap.org

แรกเป็นตัวอย่างของข้อมูลจากหน่วยงานราชการของประเทศไทย ในขณะที่แหล่งข้อมูลที่เหลือเป็นผู้ให้บริการข้อมูลสภาพอากาศทางอินเทอร์เน็ตของต่างประเทศ

งานวิจัยนี้อ่านข้อมูลพยากรณ์อากาศจากผู้ให้บริการแบบข้ามโดเมน (cross domain) ด้วยคลังโปรแกรมเคิร์ล (cURL[7]) ซึ่งได้รวบรวมชุดคำสั่งที่จำเป็นไว้ในฟังก์ชัน `get_remote_content` ดังภาพที่ 4.16 ซึ่งมีการเรียนฟังก์ชัน `curl_init` เพื่อสร้างตัวแปรหลักสำหรับการรับส่งข้อมูล และมีการกำหนดค่าตัวเลือกของการสื่อสาร ดังรายละเอียดในตารางที่ 4.2

```
1    function get_remote_content($url)
2    {
3      $ch = curl_init($url);
4      curl_setopt($ch, CURLOPT_URL, $url);
5      curl_setopt($ch, CURLOPT_HEADER, 0);
6      curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
7      curl_setopt($ch, CURLOPT_CONNECTTIMEOUT ,0);
8      curl_setopt($ch, CURLOPT_USERAGENT, $useragent);
9      curl_setopt($ch, CURLOPT_TIMEOUT, 300);
10     $output = curl_exec($ch);
11     curl_close($ch);
12     return $output;
13   }
```

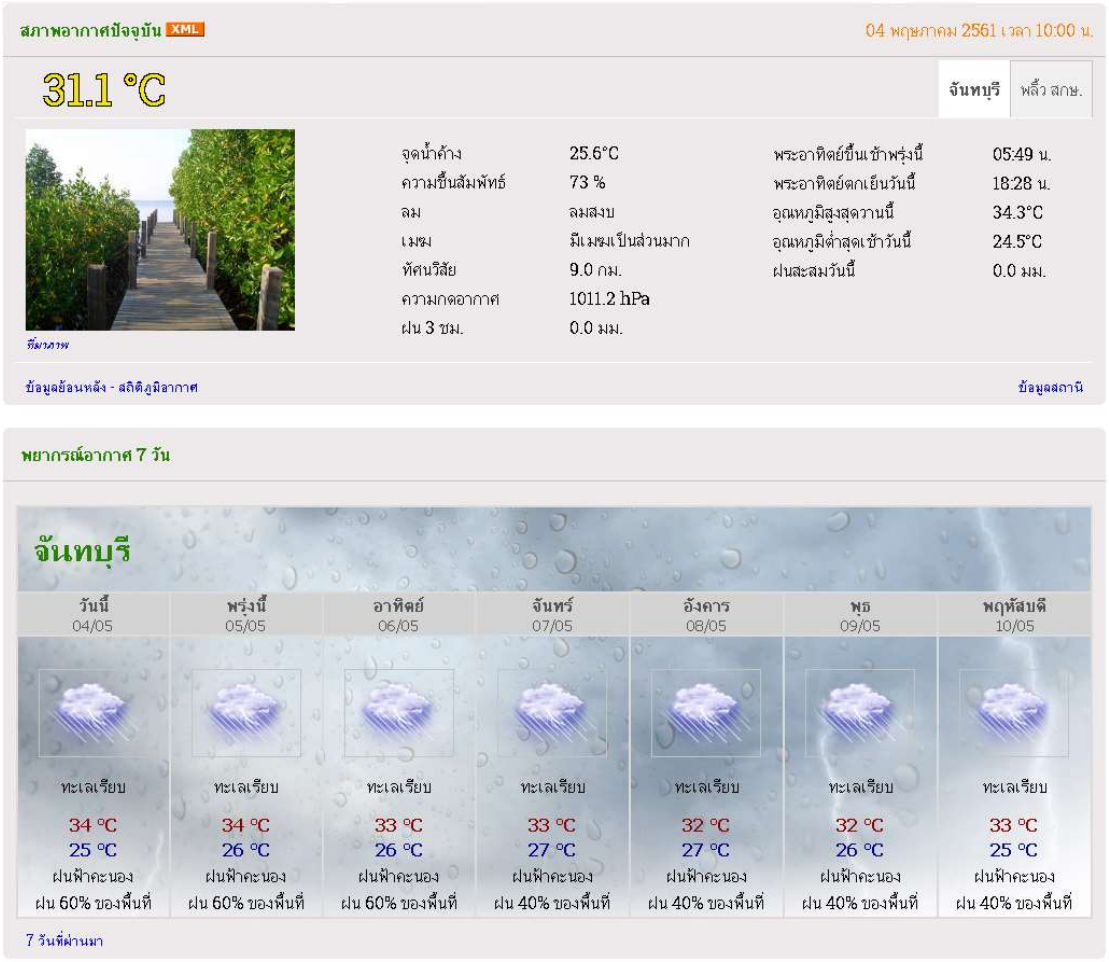
ภาพที่ 4.16 การสั่งงานคลังโปรแกรมเคิร์ล

ตารางที่ 4.2 ค่าที่ใช้ในการสั่งงานคลังโปรแกรมเคิร์ล
ที่มา : PHP Group, 2018.

ลำดับ	ค่าคงที่	ความหมาย	ค่าที่กำหนด
1	CURLOPT_URL	ยูอาร์แอลเป้าหมาย	ตามค่าที่ส่งมา
2	CURLOPT_HEADER	ข้อมูลส่วนหัว	ไม่มี ใช้ค่าโดยปริยาย
3	CURLOPT_RETURNTRANSFER	รูปแบบการนำเสนอข้อความตอบรับจากเป้าหมาย	1 (คืนค่าเป็นสายอักขระ)
4	CURLOPT_CONNECTTIMEOUT	ระยะเวลารอระหว่างการพยายามเชื่อมต่อ (วินาที)	0 (รอจนกว่าจะเชื่อมต่อได้)
5	CURLOPT_USERAGENT	รายละเอียดของเบราเซอร์	ตามผู้ใช้กำหนด
6	CURLOPT_TIMEOUT	ระยะเวลารอระหว่างการทำงาน (วินาที)	300 วินาที

7 https://curl.haxx.se/

ในกรณีของกรมอุตุนิยมวิทยานั้น มีบริการข้อมูลทั้งแบบดูผ่านทางเว็บไซต์ เช่น `https://www.tmd.go.th/province.php?StationNumber=48480` จะแสดงข้อมูลพยากรณ์อากาศของจังหวัดจันทบุรี 7 วันล่วงหน้าดังแสดงในภาพที่ 4.17



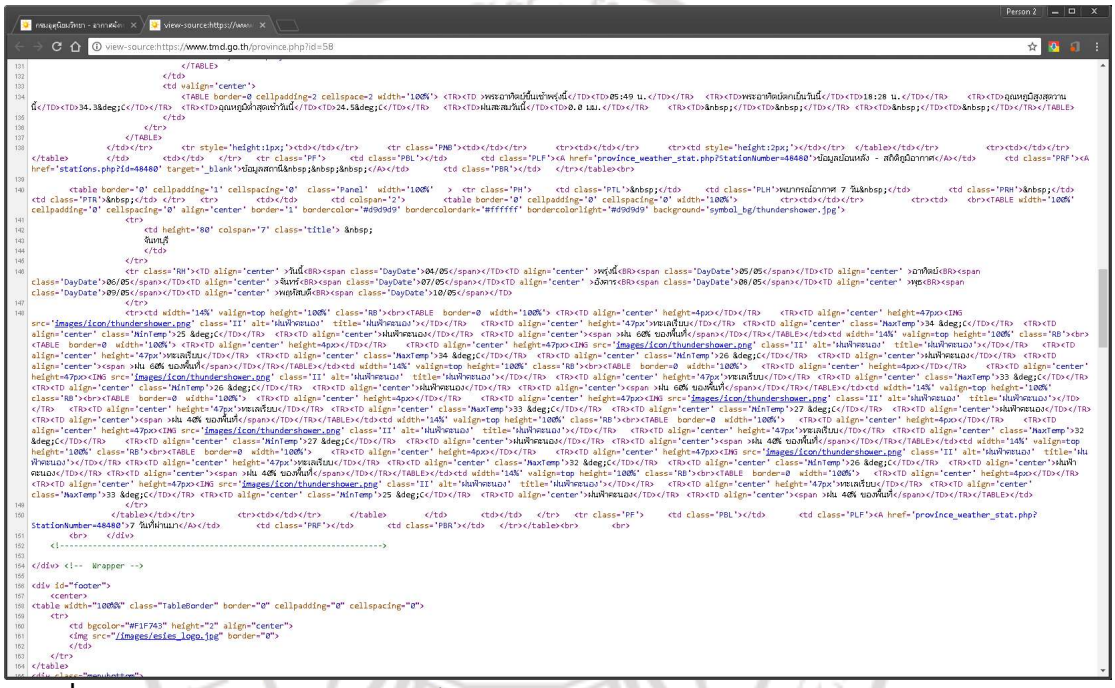
ภาพที่ 4.17 การสั่งงานคลังโปรแกรมเคิร์ล

โดยสามารถเลือกและข้อมูลของจังหวัดจันทบุรีได้จากสถานีตรวจอากาศสองสถานี คือ รหัส 48480 ซึ่งหมายถึงสถานีตรวจอากาศจันทบุรี และรหัส 48481 ซึ่งหมายถึงสถานีตรวจอากาศพลิ้ว สำหรับการบริการข้อมูลอีกแบบหนึ่งนั้นจะเป็นบริการข้อมูลผ่านเว็บ เอพีไอ[8] ซึ่งเป็นไปตามนโยบายรัฐบาลอิเล็กทรอนิกส์ (e-Government) ตามมาตรฐานข้อมูลเปิด (Open Data) ทำให้มีความสะดวกในการเรียก, การสกัดข้อมูลและการนำไปใช้แบบอัตโนมัติมากกว่ากรณีแรก

อย่างไรก็ตามเนื่องจากในช่วงเวลาที่ผู้วิจัยพัฒนาระบบนั้นการบริการข้อมูลผ่านเว็บเอพีไอของกรมอุตุนิยมวิทยายังไม่สามารถนำมาใช้กับงานวิจัยนี้ได้อย่างสะดวกนั้น ดังนั้นงานวิจัยนี้จึงจะดึงข้อมูลจากหน้าเว็บไซต์โดยตรงและใช้เทคนิคการเขียนโปรแกรมเพื่อสกัดข้อมูลที่ต้องการ ซึ่งอาจจะไม่ยืดหยุ่นเท่ากับการเรียกใช้งานผ่านเว็บเอพีไอ แต่สามารถใช้เป็นแนวทางการสกัดข้อมูล

8 http://data.tmd.go.th/api/index1.php

จากเว็บไซต์อื่นที่ไม่มีบริการเอพีไอให้โปรแกรมเรียกใช้อย่างเป็นทางการได้ โดยการอ่านข้อมูลพยากรณ์อากาศจากเว็บไซต์กรมอุตุนิยมวิทยานั้นทำด้วยการใช้คลังโปรแกรมเคิร์ลเพื่ออ่านข้อมูลพยากรณ์อากาศ 7 วันจากยูอาร์แอลที่กล่าวไปแล้วในย่อหน้าก่อนหน้านี้ โดยส่วนบนของเว็บแสดงข้อมูลปัจจุบัน และมีข้อมูลพยากรณ์อากาศล่วงหน้า 7 วันแสดงในส่วนล่าง และภาพที่ 4.18 แสดงรหัสต้นฉบับจากเว็บไซต์ดังกล่าว



ภาพที่ 4.18 ตัวอย่างรหัสต้นฉบับของเว็บพยากรณ์ข้อมูลรายจังหวัดของกรมอุตุนิยมวิทยา

```
1    function get_tmd_data($url)
2    {
3      // Get content
4      $output = get_remote_content($url);
5      ...
6      // Search for target table
7      $key = "พยากรณ์อากาศ 7 วัน";
8      $idx1 = strpos($output, $key, 0);
9      $key = "<!--------------------------------->";
10     $idx2 = strpos($output, $key, $idx1);
11     $content = substr($output, $idx1, $idx2-$idx1);
12     ...
13   }
```

ภาพที่ 4.19 การสั่งงานคลังโปรแกรมเคิร์ล

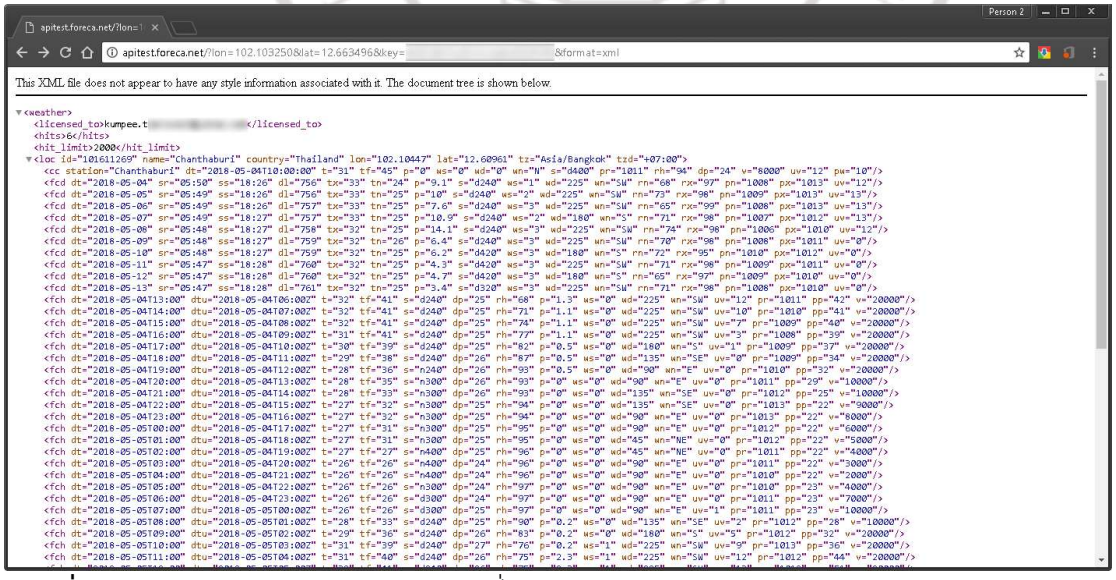
การสกัดข้อมูลทำด้วยการเรียกใช้ฟังก์ชัน `get_tmd_data` ซึ่งจะมีการเรียกฟังก์ชัน `get_remote_content` เพื่ออ่านข้อมูลพยากรณ์อากาศจากเครื่องคอมพิวเตอร์ให้บริการของกรมอุตุนิยมวิทยามาตรวจสอบหาแท็ก (tag) ที่เกี่ยวกับการพยากรณ์อากาศ 7 วัน ซึ่งจาก

การตรวจสอบ ณ ขณะที่กำลังพัฒนางานวิจัยนี้ผู้วิจัยพบว่าข้อมูลพยากรณ์อากาศ 7 วันนี้อยู่ในส่วนของแท็กตารางที่เริ่มต้นด้วยข้อความ "พยากรณ์อากาศ 7 วัน" และมีการลงข้อมูลหมายเหตุ (comment) ในส่วนท้ายตารางเป็น "<!----------->" ดังนั้นผู้วิจัยจึงกำหนดให้โปรแกรมตรวจสอบหาตำแหน่งข้อความทั้งสองจุดนี้ ดังตัวอย่างในภาพที่ 4.19 (ค่าที่กำหนดในตัวแปร `$key` ในภาพที่ 4.19 บรรทัดที่ 9 นั้นจะสั้นกว่าความเป็นจริงเพื่อประหยัดพื้นที่กระดาษ) เมื่อพบแท็กทั้งสองนี้แล้วจึงใช้เทคนิคการตรวจสอบสายอักขระเพื่อค้นหาแท็กย่อยที่เก็บข้อมูลอุณหภูมิและข้อมูลอื่นที่ต้องการ โดยการตรวจหาตำแหน่งข้อมูลที่ต้องการนั้นทำด้วยการเรียกใช้ฟังก์ชัน `strpos` เพื่อคำนวณหาตำแหน่งเริ่มต้นอักขระ ‘<’ และตำแหน่งของ ‘>’ ซึ่งแสดงจุดสิ้นสุดแท็กที่เกี่ยวข้อง เมื่อพบแล้วจึงสกัดข้อความระหว่างแท็กทั้งสองด้วยฟังก์ชัน `substr` เสร็จแล้วจึงส่งข้อความที่สกัดได้ให้ฟังก์ชัน `trim` เพื่อลบข้อความว่างหรือไม่ต้องการออกจากส่วนต้นและส่วนท้ายของข้อความ ผลลัพธ์ที่ได้คือข้อมูลที่ต้องการ ซึ่งฟังก์ชันจะส่งกลับออกไปให้ผู้ใช้งานต่อไป

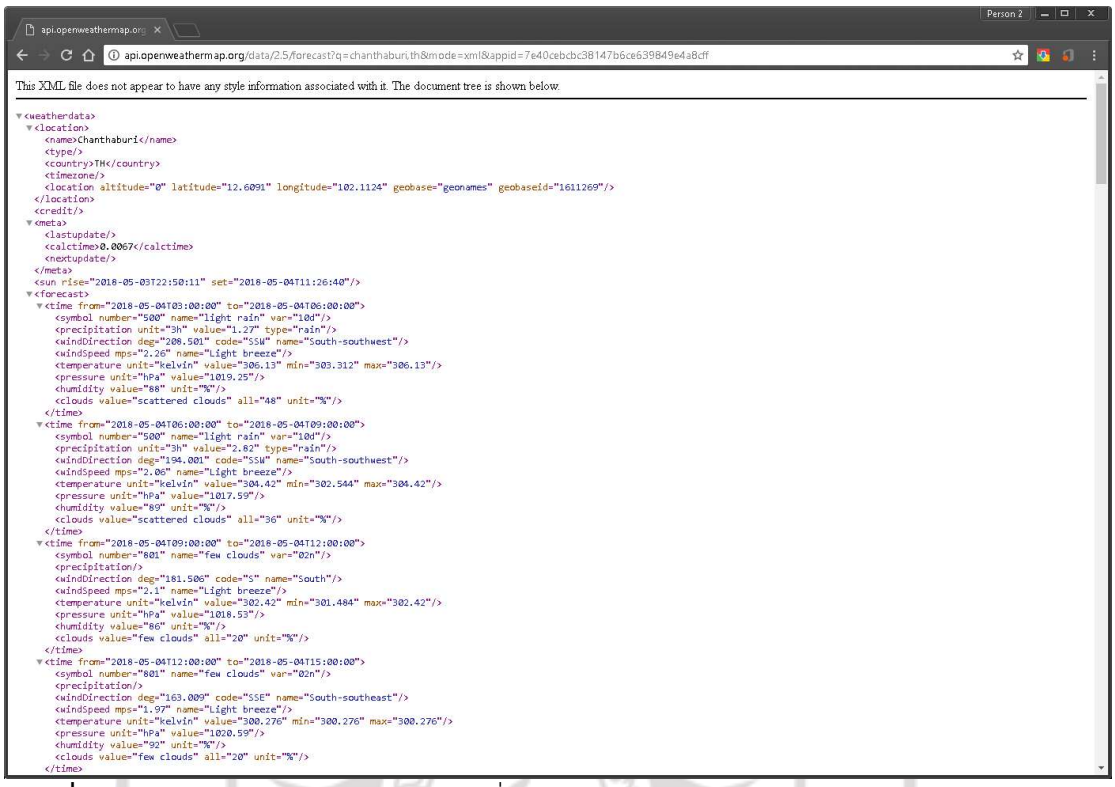
สำหรับแหล่งให้บริการข้อมูลสภาพอากาศแหล่งที่สอง คือ ฟอร์คาซึ่งมีบริการข้อมูลให้ผ่านเว็บเอพีไอ จึงทำให้การพัฒนาโปรแกรมเพื่อเรียกใช้งานทำได้สะดวกกว่าการค้นหาและสกัดสายอักขระโดยตรง โดยผู้ใช้สามารถร้องขอข้อมูลได้จากยูอาร์แอล `http://apitest.foreca.net/` และมีรูปแบบข้อความร้องขอเป็น

```
?lon=LAT&lat=LON&key=API_KEY&format=xml
```

เมื่อ `LAT` และ `LON` คือพิกัดละติจูดและลองจิจูดของตำแหน่งที่ต้องการทราบข้อมูล, `API_KEY` คือคีย์หรือรหัสผู้ใช้สำหรับการเรียกใช้ โดยในงานวิจัยนี้กำหนดให้มีการคืนผลลัพธ์เป็น `xml` ซึ่งหมายถึงการคืนข้อมูลในรูปแบบเอกสารเอ็กซ์เอ็มแอล ภาพที่ 4.20 แสดงตัวอย่างเอกสารเอ็กซ์เอ็มแอลที่ส่งกลับมาจากฟอร์คา



ภาพที่ 4.20 ตัวอย่างข้อมูลพยากรณ์อากาศที่ได้จากฟอร์คา



ภาพที่ 4.21 ตัวอย่างข้อมูลพยากรณ์อากาศที่ได้จากโอเพ่นเวเธอร์แมพ

แหล่งข้อมูลสุดท้ายคือโอเพ่นเวเธอร์แมพซึ่งมีบริการผ่านทางเอพีไอเช่นเดียวกัน โดยสามารถร้องขอข้อมูลจากยูอาร์แอล http://api.openweathermap.org/data/2.5/ และมีรูปแบบข้อความร้องขอเป็น

```
forecast?q=LOCATION&mode=xml&appid=API_KEY
```

เมื่อ `LOCATION` คือชื่อสถานที่ที่ต้องการทราบข้อมูลในรูปแบบของชื่อพื้นที่คั่นด้วยเครื่องหมายลูกน้ำต่อด้วยชื่อประเทศ เช่น `chanthaburi,th` และ `API_KEY` คือคีย์หรือรหัสผู้ใช้สำหรับการเรียกใช้ ดังภาพที่ 4.21

โดยปกติแล้วข้อมูลจากแหล่งที่ให้บริการแบบเอพีไอนั้นจะมีรูปแบบข้อมูลที่หลากหลาย เช่น เจสัน (JavaScript Object Notation : JSON), จีโอเจสัน (Geographic JSON : GeoJSON) หรือรูปแบบอื่นๆ สำหรับงานวิจัยนี้สนใจรูปแบบเอกสารเอ็กซ์เอ็มแอล ซึ่งมีรูปแบบการจัดเรียงข้อมูลภายในเป็นไปตามแบบจำลองวัตถุเอกสารหรือดีโอเอ็ม (Document Object Model : DOM) ดังนั้นการสกัดข้อมูลในภาษาพีเอชอีนั้นทำได้ด้วยการเรียกใช้เมท็อดต่าง ๆ จากคลาส `DOMDocument` ยกตัวอย่างเช่นการค้นหาแท็กที่ต้องการนั้นทำด้วยการเรียกใช้เมท็อด `getElementsByTagName` โดยส่งชื่อแท็กเป็นอาร์กิวเม้นต์ เมื่อพบจึงเรียกใช้เมท็อด `getAttribute` เพื่ออ่านข้อมูลที่ต้องการ โปรแกรมจะแปลงข้อมูลตอบรับจากเครื่องคอมพิวเตอร์ให้บริการบันทึกใส่ในตัวแปร `output` ให้เป็นโครงสร้างเอกสารดีโอเอ็ม แล้วเก็บในตัวแปร `xml` จากนั้นจึงค้นหาแท็ก `fcd` แล้วเก็บโหนดของแท็กนี้รวมทั้งโหนดลูกทั้งหมดของแท็กนี้ในตัวแปร

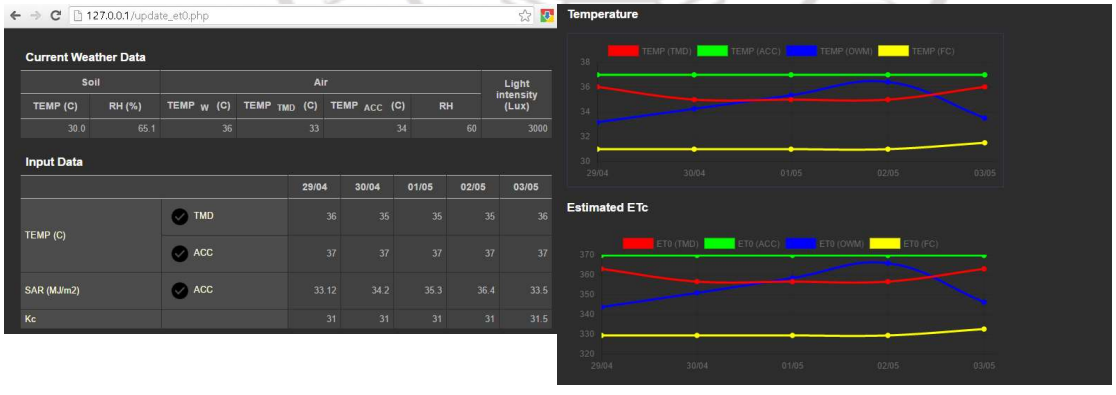
nodes จากนั้นจึงวนเข้าไปอ่านค่าจากตัวแปร nodes ดังชุดคำสั่งในภาพที่ 4.22 บรรทัดที่ 7 - 10 ซึ่งเป็นตัวอย่างการอ่านข้อมูลอุณหภูมิต่ำสุด, อุณหภูมิสูงสุด, ปริมาณฝนตกต่ำสุด และฝนตกสูงสุด ตามลำดับ

```
1    ...
2    $xml = DOMDocument::load($output);
3    $nodes = $xml->getElementsByTagName('fcd');
4    ...
5    foreach($nodes as $node) {
6      ...
7      $temp_min = $node->getAttribute('tn');
8      $temp_max = $node->getAttribute('tx');
9      $rain_min = $node->getAttribute('rn');
10     $rain_max = $node->getAttribute('rx');
11     ...
12   }
13   ...
```

ภาพที่ 4.22 การค้นหาข้อมูลที่ต้องการจากแท็กในเอกสารเอ็กซ์เอ็มแอล

2) ส่วนติดต่อผู้ใช้แบบกราฟิกส์

ข้อมูลที่เก็บในเครื่องให้บริการนั้นสามารถนำออกมาแสดงเป็นภาพกราฟิกส์หรือแผนภูมิได้ โดยงานวิจัยนี้ออกแบบให้ระบบนำข้อมูลอากาศที่วัดได้จากสถานีตรวจอากาศของระบบเปรียบเทียบกับข้อมูลพยากรณ์อากาศล่วงหน้า 5 วันจากแหล่งให้บริการข้อมูลต่าง ๆ ดังภาพที่ 4.23 ซึ่งแสดงเว็บแอปพลิเคชันต้นแบบข้อมูลปัจจุบันในส่วนข้อมูลอากาศปัจจุบัน (Current Weather Data) ซึ่งอยู่ส่วนบนของภาพ



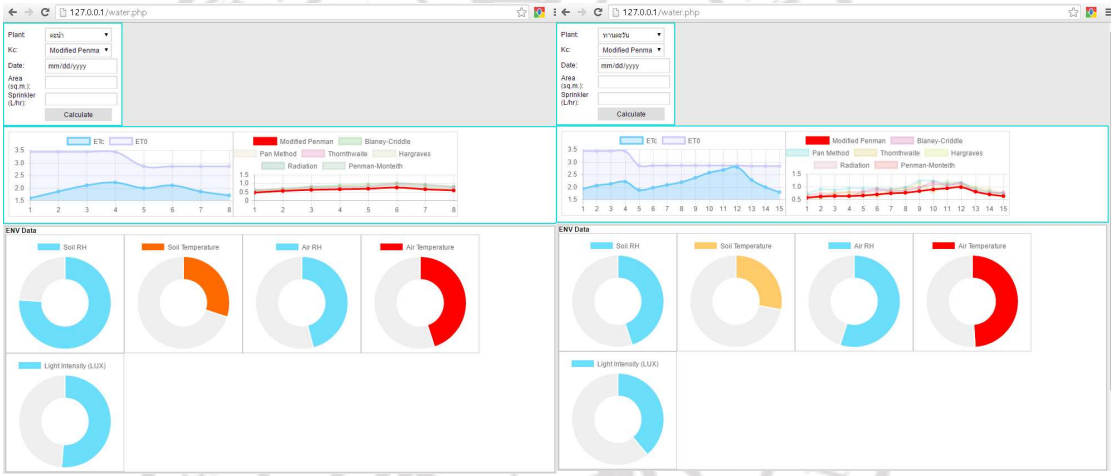
(ก) (ข)

ภาพที่ 4.23 เว็บแอปพลิเคชั่นต้นแบบสำหรับแสดงข้อมูลอากาศปัจจุบัน

ข้อมูลส่วนนี้ประกอบอุณหภูมิของดิน (คอลัมน์ TEMP (C)), ความชื้นสัมพัทธ์ของดิน (คอลัมน์ RH (%)), อุณหภูมิของอากาศ (คอลัมน์ $TEMP_W$ (C)) และความเข้มแสงแวดล้อม ซึ่งเป็นค่าที่ได้โหนดรับบริการ ส่วนข้อมูลอุณหภูมิ $TEMP_{TMD}$ และ $TEMP_{ACC}$ เป็นข้อมูลจากกรม

อุตุนิยมวิทยาและโอเพ่นเวเตอร์แมพตามลำดับ สำหรับข้อมูลส่วนล่างของภาพที่ 4.23 (ก) นั้นเป็นการเปรียบเทียบข้อมูลพยากรณ์อากาศจากแหล่งต่าง ๆ เพื่อให้ผู้ใช้ตัดสินใจว่าจะเลือกใช้ข้อมูลใด และภาพที่ 4.23 (ข) แสดงส่วนติดต่อผู้ใช้อีกแบบหนึ่งซึ่งข้อมูลเปรียบเทียบค่าอุณหภูมิและค่าความต้องการใช้น้ำของพืชที่คำนวณได้ด้วยการใช้ค่าความต้องการใช้น้ำอ้างอิงจากข้อมูลต่าง ๆ

ภาพที่ 4.24 (ก) แสดงส่วนติดต่อผู้ใช้ที่นำข้อมูลต่าง ๆ มารวมกันแล้ว โดยส่วนบนแสดงตัวเลือกสำหรับการคำนวณ ซึ่งระบบจะแสดงข้อมูลความต้องการของพืชและข้อมูลความต้องการน้ำของพืชอ้างอิงที่คำนวณได้ให้ทราบในรูปแบบของแผนภูมิเส้น รวมทั้งมีการเปรียบเทียบค่าความต้องการใช้น้ำของพืชที่ได้จากเทคนิคต่าง ๆ เปรียบเทียบให้เห็นด้วย สำหรับส่วนล่างจะเป็นข้อมูลอากาศปัจจุบันจากโหนดรับบริการ



(ก) (ข)

ภาพที่ 4.24 เว็บแอปพลิเคชั่นสำหรับการแสดงข้อมูลความต้องการใช้น้ำของพืช

ในการพัฒนาระบบงานส่วนที่เป็นบริการเว็บนั้นผู้วิจัยไม่ได้ใช้เฟรมเวิร์คใด ๆ ลดขนาดโปรแกรมบนเว็บให้มีขนาดเล็กที่สุด อย่างไรก็ตามในการพัฒนาส่วนติดต่อผู้ใช้ที่จะทำงานบนสมาร์ทโฟนนั้นผู้วิจัยได้เลือกใช้เฟรมเวิร์คสำหรับการพัฒนาโปรแกรมบนสมาร์ทโฟน เนื่องจากชุดคำสั่งที่เขียนได้นั้นสามารถนำไปแปลงและสร้างเป็นโปรแกรมหรือแอป (app) ที่สามารถทำงานบนระบบปฏิบัติการของสมาร์ทโฟนแต่ละค่ายได้ทั้งระบบปฏิบัติการแอนดรอย์ด (Android) และระบบปฏิบัติการไอโอเอส (iOS)

ดังนั้นในงานวิจัยนี้พัฒนาโปรแกรมบนสมาร์ทโฟนด้วยการใช้แองกูลาร์เจเอส[9] (AngularJS) ซึ่งเป็นเครื่องมือแบบรหัสเปิดพัฒนาโดยกูเกิล มีพื้นฐานจากภาษาจาวาสคริปต์และไทป์สคริปต์ (TypeScript) สำหรับการพัฒนาโปรแกรมประยุกต์บนเว็บ ซึ่งสามารถนำชุดคำสั่งที่ได้ไปใช้งานร่วมกับอาปาเช่คอร์โดวา[10] (Apache Cordova) เพื่อสร้างเป็นแอปบนสมาร์ทโฟนได้

9 https://angular.io/

10 https://cordova.apache.org/

3) ส่วนติดต่อผู้ใช้เพื่อการคำนวณความต้องการใช้น้ำ

ผู้ใช้สามารถคำนวณความต้องการใช้น้ำของพืชได้โดยการระบุข้อมูลที่จำเป็น ซึ่งประกอบด้วย จังหวัด, ชนิดพืช, วันที่, ขนาดพื้นที่ และเทคนิคการคำนวณ เป็นต้น โดยผู้ใช้สามารถเลือกพารามิเตอร์เหล่านี้จากเครื่องมือดรอปดาวน์ลิสต์ (dropdown list) ในหน้าหลักของโปรแกรม โดยโปรแกรมจะส่งค่าที่ผู้ใช้เหล่านี้ไปให้คอมพิวเตอร์ให้บริการด้วยวิธีเอชทีทีพีโพสต์ (HTTP POST) ตารางที่ 4.3 แสดงพารามิเตอร์ต่าง ๆ และตัวอย่างค่าที่เป็นไปได้

ตารางที่ 4.3 การร้องขอข้อมูลแผนการให้น้ำ

ชื่อตัวแปร	ความหมาย	ค่าที่เป็นไปได้
PROV_ID	ตัวเลข 2 หลักแสดงรหัสจังหวัด	ตามมาตรฐาน ISO 3166-2:TH
VEG_ID	รหัสพืช	1: คะน้า
MONTH_S	เดือนที่เริ่มต้นปลูก	[1, 12]
KC_METHOD	วิธีการคำนวณ	mp: Modified Penman bc: Blaney-Criddle p: Pan Method t: Thornthwaite h: Hargraves r: Radiation pm: Penman-Monteith

จากนั้นเครื่องคอมพิวเตอร์ให้บริการจะสกัดข้อมูลต่าง ๆ แล้วทำการคำนวณความต้องการใช้น้ำของพืชแล้วสร้างเป็นแผนการให้น้ำโดยใช้เทคนิคที่กล่าวถึงในหัวข้อ 3.5.2 เสร็จแล้วจึงส่งผลกลับไปให้ผู้ใช้ โดยข้อมูลที่คืนกลับมาจะประกอบด้วยปริมาตรน้ำ (ลูกบาศก์เซนติเมตร), ระยะเวลา (วินาที) และค่าน้ำโดยประมาณต่อการให้น้ำหนึ่งครั้ง (บาท) โดยงานวิจัยนี้ใช้อัตราค่าน้ำประมาณจากการประปาส่วนภูมิภาค (2560) ดังแสดงในตารางที่ 4.4 ซึ่งในงานวิจัยนี้จะคำนวณค่าน้ำจากอัตราค่าน้ำสำหรับผู้ใช้ประเภทที่อยู่อาศัยเป็นหลัก

ตารางที่ 4.5 แสดงข้อมูลทดสอบการคำนวณแผนการให้น้ำและระยะเวลาการให้น้ำ โดยผู้วิจัยกำหนดพื้นที่ตัวอย่างเป็นจังหวัดจันทบุรี, กำหนดชนิดพืชเป็นคะน้า, กำหนดวันที่เริ่มต้นการปลูกเป็นวันที่ 11 มกราคม 2560, ขนาดพื้นที่ และกำหนดเทคนิคการคำนวณเป็นวิธี Modified Penman โดยมีระยะเวลาทดลองตามกำหนดเท่ากับ 60 วัน

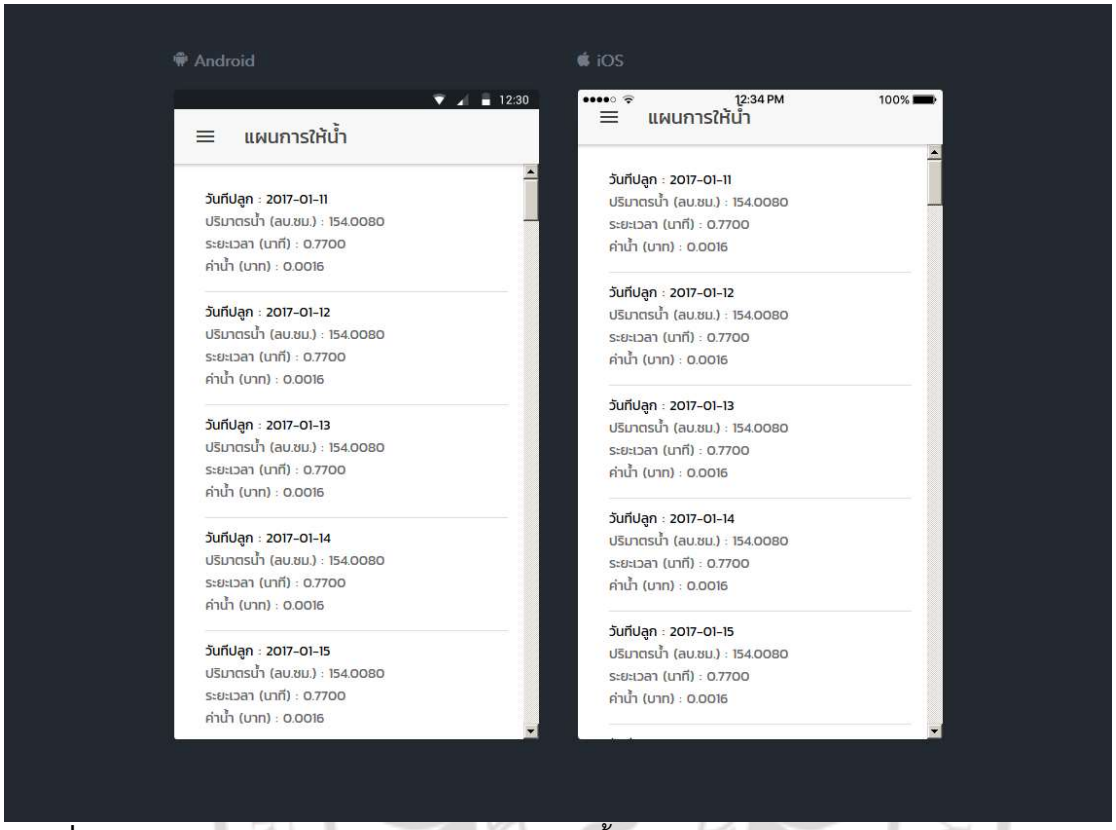
ตารางที่ 4.4 อัตราค่าน้ำประปาสาขาอื่นทั่วไประเทศ ประเภทที่อยู่อาศัย
ที่มา: การประปาส่วนภูมิภาค, 2560

ช่วงการใช้น้ำ	จำนวนหน่วย	ที่อยู่อาศัย		
ค่าน้ำขั้นต่ำ		ราคา	เป็นเงิน	รวมเงิน
0 - 10	10	10.20	102.00	102.00
11 - 20	10	16.00	160.00	262.00
21 - 30	10	19.00	190.00	452.00
31 - 50	10	21.20	424.00	876.00

ตารางที่ 4.5 ตัวอย่างแผนการให้น้ำสำหรับคะน้า

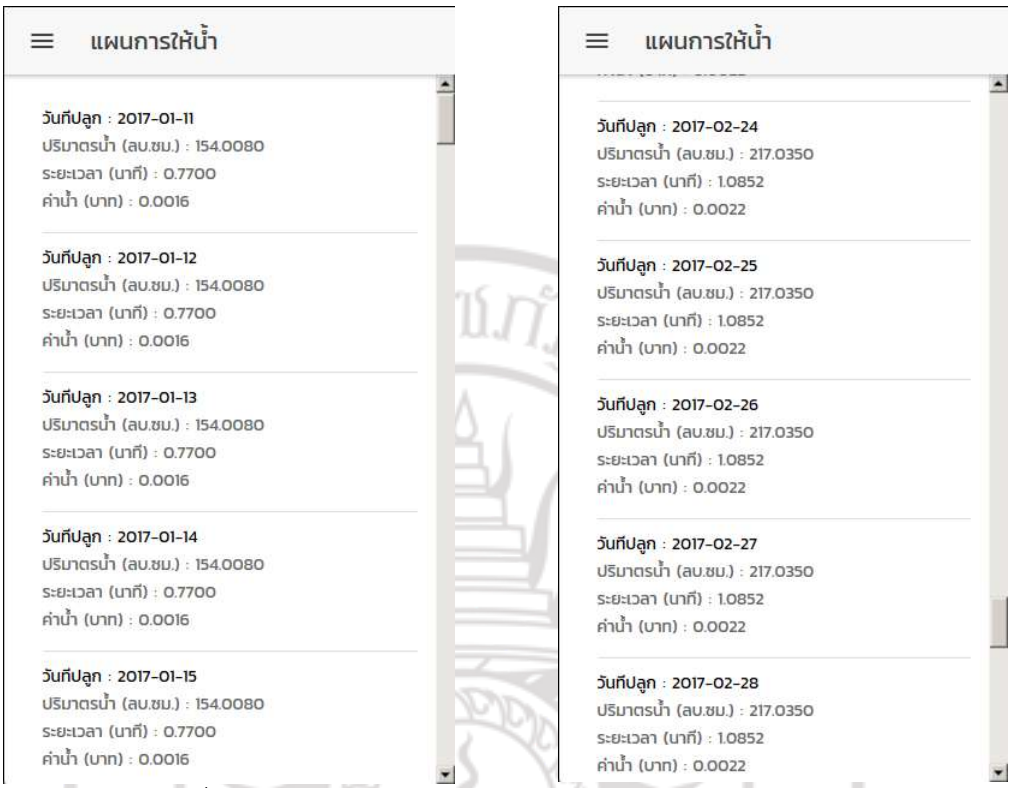
ลำดับ	วันที่	ความต้องการน้ำ (มิลลิเมตร/วัน)	ปริมาตร (ลูกบากศ์เซนติเมตร)	ระยะเวลา (นาที)	ค่าน้ำ (บาท)
1	2017-01-11	1.7112	154.0080	0.7700	0.0016
2	2017-01-12	1.7112	154.0080	0.7700	0.0016
3	2017-01-13	1.7112	154.0080	0.7700	0.0016
4	2017-01-14	1.7112	154.0080	0.7700	0.0016
5	2017-01-15	1.7112	154.0080	0.7700	0.0016
6	2017-01-16	1.7112	154.0080	0.7700	0.0016
7	2017-01-17	1.7112	154.0080	0.7700	0.0016
8	2017-01-18	2.0088	180.7920	0.9040	0.0018
9	2017-01-19	2.0088	180.7920	0.9040	0.0018
...	...	...	...	...	...

ภาพที่ 4.25 แสดงตัวอย่างหน้าจอการแสดงแผนการให้น้ำ โดยใช้ข้อมูลจากการเพาะปลูกในหัวข้อที่ผ่านมา ซึ่งในภาพได้แสดงทั้งการทำงานบนระบบปฏิบัติการแอนดรอย์ดและระบบปฏิบัติการไอโอเอส



ภาพที่ 4.25 แสดงทดสอบการแสดงข้อมูลแผนการให้น้ำบนโปรแกรมจำลอง

ภาพที่ 4.26 แสดงรายละเอียดของข้อมูลที่ช่วงเวลาการปลูกต่าง ๆ กัน ในภาพที่ 4.26 (ก) นั้นแสดงข้อมูลช่วงเริ่มต้นเพาะปลูกซึ่งอยู่ในช่วงประมาณกลางเดือนมกราคม 2560 ซึ่งระบบคำนวณความต้องการน้ำของคะน้าประมาณวันละ 154 ลูกบาศก์เซนติเมตร และได้ประมาณเวลาที่ต้องใช้ในการให้น้ำจากข้อมูลประสิทธิภาพของวาล์วน้ำพบว่าต้องเปิดน้ำเป็นเวลาประมาณ 0.77 นาทีหรือประมาณ 47 วินาที และมีค่าน้ำในช่วงนี้ประมาณวันละ 0.0016 บาท สำหรับภาพที่ 4.26 (ข) เป็นข้อมูลแผนการให้น้ำในช่วงท้ายของการปลูก ประมาณวันที่ 55 - 59 ซึ่งในช่วงนี้ระบบคำนวณความต้องการน้ำวันละ 217 ลูกบาศก์เซนติเมตร โดยทำได้ด้วยการเปิดวาล์วน้ำเป็นระยะเวลาประมาณ 1 นาที



(ก) เริ่มต้นปลูก (ข) ช่วงท้ายของการปลูก

ภาพที่ 4.26 รายละเอียดแผนการให้น้ำช่วงเวลาต่าง ๆ

4.2.5 การทดลอง

การทดสอบปลูกคะน้าทำตามรายละเอียดที่กล่าวในบทที่ 3 โดยมีการเตรียมพื้นที่ทดลองในช่วงต้นเดือนมกราคม 2560 โดยภาพที่ 4.27 แสดงขั้นตอนการเตรียมอุปกรณ์ต่าง ๆ

(ก) (ข) (ค)

(ง) (จ) (ฉ)

ภาพที่ 4.27 การเตรียมการเพาะปลูก

ภาพที่ 4.27 (ก) แสดงการเตรียมท่อพีวีซีขนาด 1/2 นิ้ว ซึ่งได้นำมาตัดแบ่งและเจาะรูสำหรับการติดวาล์วน้ำขนาดเล็กดังภาพที่ 4.27 (ข) และภาพที่ 4.27 (ค) หลังจากนั้นจึงทดลองเตรียมข้อต่อระหว่างแนวส่งน้ำต่าง ๆ ดังภาพที่ 4.27 (ง) และภาพที่ 4.27 (จ) เมื่อเสร็จแล้วจึงเตรียมดินและใส่เมล็ดพันธุ์ลงในกระถางดังภาพที่ 4.27 (ฉ) ซึ่งผู้วิจัยนำแผนการให้น้ำที่คำนวณได้มาทดลองโดยได้มีการบันทึกภาพระหว่างการทดลองไว้เป็นระยะ โดยตารางที่ 4.4 แสดงวันที่ที่มีการบันทึกภาพการเจริญเติบโตของพืช

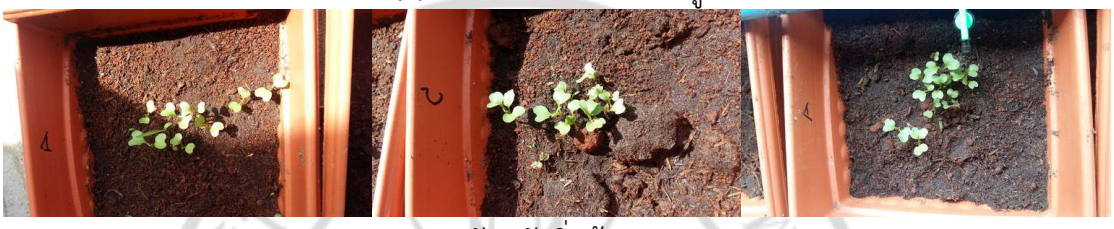
ตารางที่ 4.6 วันที่มีการบันทึกภาพ เริ่มต้นเพาะปลูกวันที่ 11 มกราคม 2560

มกราคม							**กุมภาพันธ์**							**มีนาคม**						
อา	จ	อ	พ	พฤ	ศ	ส	อา	จ	อ	พ	พฤ	ศ	ส	อา	จ	อ	พ	พฤ	ศ	ส
1	2	3	4	5	6	7				1	2	3	4				1	2	3	4
8	9	10	11	12	13	14	5	6	7	8	9	10	11	5	6	7	8	9	10	11
15	16	17	18	19	20	21	12	13	14	15	16	17	18	12	13	14	15	16	17	18
22	23	24	25	26	27	28	19	20	21	22	23	24	25	19	20	21	22	23	24	25
29	30	31					26	27	28					26	27	28	29	30	31	

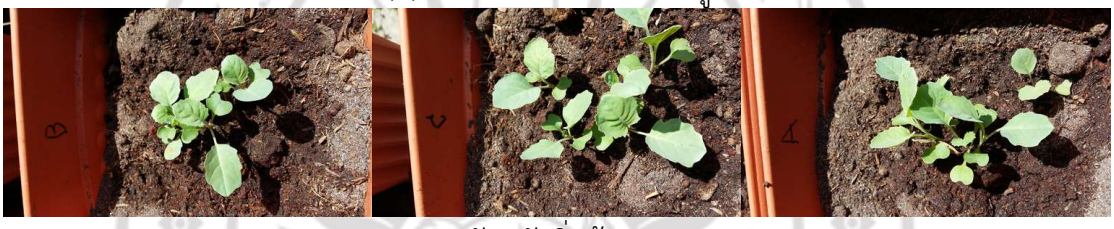
ซึ่งข้อมูลแผนการให้น้ำที่ระบบสร้างได้นั้นมีระยะเวลาการให้น้ำเฉลี่ยประมาณ 205 ลูกบาศก์เซนติเมตรต่อวัน ใช้น้ำตลอดช่วงการทดลองประมาณ 12,357 ลูกบาศก์เซนติเมตรหรือประมาณ 12.35 ลิตรตลอดการทดลองเป็นเวลา 60 วัน ซึ่งหากคิดค่าน้ำจากอัตราค่าน้ำประปาประเภทที่อยู่อาศัยในช่วงการใช้น้ำไม่เกิน 10 ลูกบาศก์เมตรต่อเดือนจะมีอัตราค่าน้ำประปาเท่ากับ 10.20 บาทต่อลูกบาศก์เมตร (การประปาส่วนภูมิภาค, 2560) และจากแผนการให้น้ำนี้จะคิดเป็นค่าน้ำประมาณ 0.06 บาทต่อเดือนหรือประมาณ 0.13 บาทต่อสองเดือน ทั้งนี้ค่าน้ำนี้ยังไม่รวมค่าบริการทั่วไปและค่าภาษีมูลค่าเพิ่ม ซึ่งจากตัวอย่างการใช้น้ำ 0.06 บาทต่อเดือนนั้นเมื่อรวมค่าบริการทั่วไป 350.00 บาทและค่าภาษีมูลค่าเพิ่ม 7% ซึ่งมีค่าประมาณ 24.50 บาท โดยไม่คิดค่าน้ำจากการใช้งานอื่นอีก จะได้ค่าน้ำต่อเดือนเท่ากับ 374.57 บาท (การประปาส่วนภูมิภาค, 2560) และภาพที่ 4.28 แสดงตัวอย่างการเจริญเติบโตของพืชในบางช่วงเวลา

(ก) 2 วันหลังเริ่มต้นเพาะปลูก

(ข) 7 วันหลังเริ่มต้นเพาะปลูก

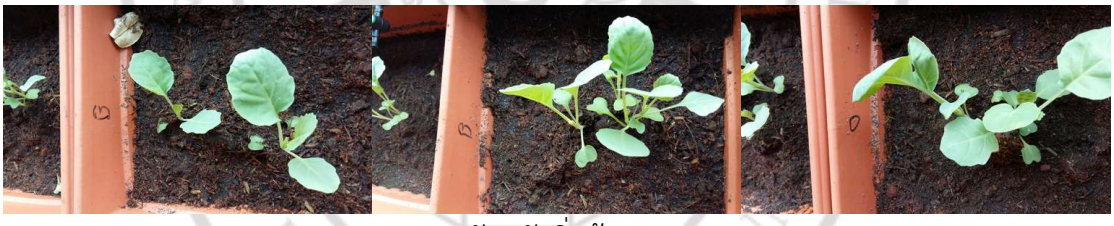


(ค) 10 วันหลังเริ่มต้นเพาะปลูก



(ค) 17 วันหลังเริ่มต้นเพาะปลูก

ภาพที่ 4.28 สัปดาห์ที่ 1-3



(ก) 21 วันหลังเริ่มต้นเพาะปลูก

(ข) 29 วันหลังเริ่มต้นเพาะปลูก

(ค) 34 วันหลังเริ่มต้นเพาะปลูก

ภาพที่ 4.29 สัปดาห์ที่ 4-7

(ก) 46 วันหลังเริ่มต้นเพาะปลูก

(ข) 50 วันหลังเริ่มต้นเพาะปลูก

ภาพที่ 4.30 สัปดาห์ที่ 8-10

จากภาพจะเห็นได้ว่าพืชมีการเจริญเติบโตเป็นไปตามที่ต้องการ และแผนการให้น้ำที่คำนวณได้นั้นนำมาใช้ได้เป็นอย่างดีดังแสดงภาพสรุปการเจริญเติบโตของการทดลองในภาพที่ 4.

ภาพที่ 4.31 ภาพสรุปการเจริญเติบโตของคะน้าระยะเวลาต่าง ๆ

4.3 อภิปรายผลการทดลอง

ระบบที่พัฒนาสามารถทำงานได้ตามต้องการ อย่างไรก็ตามประสิทธิภาพการทำงานหลักของระบบไร้สายนี้ส่วนหนึ่งคือการใช้พลังงาน ซึ่งผู้วิจัยได้ทดสอบประสิทธิภาพการใช้พลังงานดังนี้

4.3.1 การใช้พลังงาน

ในการทดสอบเพื่อการประมาณการใช้พลังงานของวงจรที่สร้างได้นั้น ผู้วิจัยได้ออกแบบนำวงจรที่ทำงานได้แต่ยังไม่ได้ต่อเซ็นเซอร์ใด ๆ มาสั่งให้ทำงานเป็นระยะเวลา 4 วินาที และเข้าสู่โหมดประหยัดพลังงานเป็นเวลา 5 วินาที หรือประมาณร้อยละ 50 แล้วทำการวัดกระแส จากนั้นผู้วิจัยได้นำบอร์ดที่ออกแบบไปต่อกับแบตเตอรีแบบลิเธียมโพลิเมอร์ แรงดัน3.7 โวลต์ 1,100 มิลลิแอมป์ชั่วโมง โดยไม่ได้ต่อแผงเซลล์สุริยะ ปล่อยให้ทำงานตามที่อธิบายไว้ในส่วนต้นของบทนี้เป็นเวลาประมาณ 28 ชั่วโมง ได้ผลลัพธ์ดังแสดงในตารางที่ 4.7 ซึ่งพบว่าขณะทำงานจะใช้กระแสโดยเฉลี่ยประมาณ 40 มิลลิแอมป์ และขณะหลับจะใช้กระแสโดยเฉลี่ยประมาณ 0.9 มิลลิแอมป์

ข้อมูลการทดสอบนี้แสดงอยู่ในภาพที่ 4.32 โดยแสดงด้วยเส้นประละเอียดสีดำ ซึ่งเมื่อพิจารณาแล้วจะเห็นว่าประจุในแบตเตอรีนั้นลดลงอย่างรวดเร็วเมื่อเวลาผ่านไปเพียงประมาณ 1 วันเท่านั้น โดยแบตเตอรีมีแรงดันไฟเหลือเพียง 3.47 โวลต์ ซึ่งแม้ว่าการทดลองนี้จะมีการประหยัดพลังงานแล้วก็ตาม แต่การที่อัตราส่วนระหว่างการหลับและตื่นนั้นใกล้เคียงกันมาก (4:5) ทำให้ระบบยังมีการใช้พลังงานมากกว่าที่ควรจะเป็น

ตารางที่ 4.7 การทดสอบการใช้พลังงานของโหนดรับบริการ (ไม่ต่อเซ็นเซอร์)

วันที่	เวลาที่วัดข้อมูล	แรงดัน (โวลต์)	จำนวนชั่วโมงที่ผ่านไป
02/11/16	05:31:00 PM	4.16	-
02/12/16	07:36:00 AM	3.85	14
02/12/16	12:35:00 PM	3.79	19
02/12/16	04:28:00 PM	3.77	23
02/1316	09:11:00 AM	3.47	28

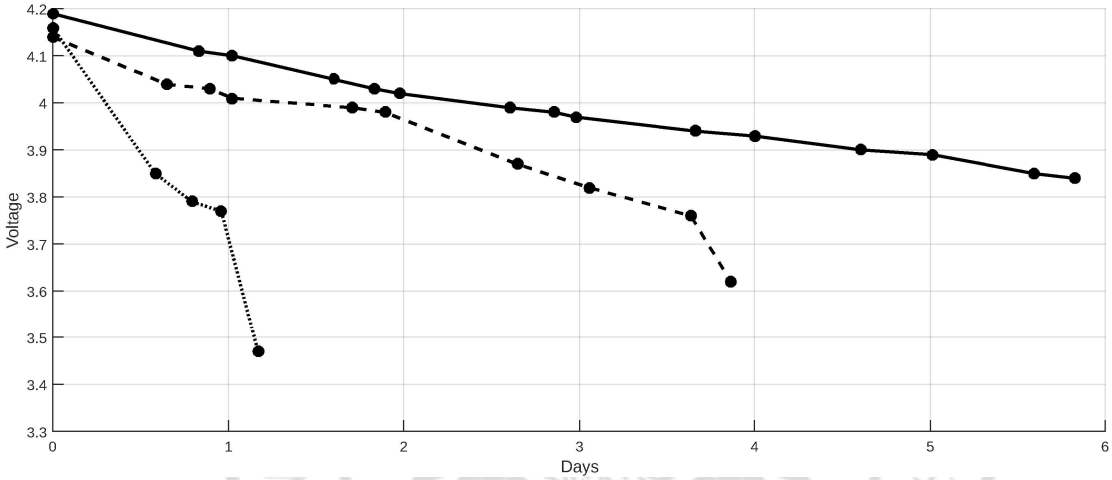
ตารางที่ 4.8 การใช้พลังงานของบอร์ด

วันที่	เวลาที่วัดข้อมูล	แรงดัน (โวลต์)	จำนวนชั่วโมงที่ผ่านไป
02/13/16	12:53:00 PM	4.19	0.0
02/14/16	09:01:00 AM	4.11	20.0
02/14/16	01:30:00 PM	4.10	24.5
02/15/16	09:30:00 AM	4.05	38.5
02/15/16	03:00:00 PM	4.03	44.0
02/15/16	06:30:00 PM	4.02	47.5
02/16/16	09:10:00 AM	3.99	62.5
02/16/16	03:00:00 PM	3.98	68.5
02/16/16	06:00:00 PM	3.97	71.5
02/17/16	10:40:00 AM	3.94	88.0
02/17/16	06:30:00 PM	3.93	96.0
02/18/16	09:00:00 AM	3.90	110.5
02/18/16	06:50:00 PM	3.89	120.3
02/17/16	08:10:00 AM	3.85	134.3
02/17/16	01:30:00 PM	3.84	139.8

ดังนั้นผู้วิจัยจึงได้ทำการทดลองอีกครั้งด้วยการปรับให้บอร์ดทำงานเป็นระยะเวลา 1 วินาทีและหลับเป็นระยะเวลา 16 วินาที แล้วทดสอบการทำงานเป็นเวลาประมาณ 7 วัน ได้ผลการใช้พลังงานดังแสดงด้วยเส้นสีดำทึบในภาพที่ 4.32 และตารางที่ 4.8 ซึ่งเมื่อพิจารณาจากภาพและตารางจะเห็นได้ว่าในกรณีนี้บอร์ดใช้พลังงานลดลงเมื่อเทียบกับกรณีแรก แบตเตอรีลดลงช้ากว่า และเมื่อเวลาผ่านไปประมาณ 6 วันนั้นแบตเตอรียังคงมีแรงดันไฟอยู่ถึง 3.84 โวลต์ ซึ่งสูงกว่าการทดลองครั้งแรกเป็นอย่างมาก และเมื่อคำนวณแล้วพบว่าอัตราส่วนการทำงานเช่นนี้ทำให้สามารถทำงานได้

ประมาณ 475 ชั่วโมงหรือประมาณ 19 วัน โดยคิดระดับวิกฤตที่ 3.0 โวลต์

อย่างไรก็ตามผลการทดสอบที่กล่าวมาในย่อหน้าข้างบนนี้เป็นการทดลองที่ไม่มีการเชื่อมต่อเซ็นเซอร์ ดังนั้นผู้วิจัยจึงทำการทดสอบต่อด้วยการต่ออุปกรณ์ทั้งหมด (ยกเว้นเครื่องชาร์จประจุและแผงเซลล์สุริยะ) กำหนดอัตราส่วนการทำงานเป็น 1:16 เช่นเดียวกับกรณีข้างบนแล้วปล่อยให้ทำงานทิ้งไว้ ได้ผลลัพธ์ดังแสดงด้วยเส้นประหยาบสีดำในภาพที่ 4.32 ผลการทดสอบพบว่าเมื่ออุปกรณ์อยู่ในสถานะตื่นจะใช้กระแสประมาณ 65.2 มิลลิแอมป์ และหากอยู่ในสถานะประหยัดพลังงานจะใช้กระแสประมาณ 1.3 มิลลิแอมป์ ซึ่งมีค่าเฉลี่ยการใช้กระแสระหว่างการทำงานแต่ละรอบประมาณ 2.85 มิลลิแอมป์ และสามารถทำงานได้ประมาณ 9 วัน



ภาพที่ 4.32 เปรียบเทียบการใช้พลังงาน

จากข้อมูลดังกล่าวแสดงให้เห็นว่าการประหยัดพลังงานนี้ได้ผลดีมาก เนื่องจากการทดลองนี้ใช้อัตราส่วนการตื่นและหลับเท่ากับ 1:16 ซึ่งผลที่ผ่านมาสามารถทำงานได้เป็นเวลาประมาณ 9 วัน แต่สำหรับการใช้งานจริงนั้นมีอัตราส่วนการตื่นและหลับสูงมากกว่านี้มาก จากการสังเกตโดยแล้วพบว่าการวัดและส่งข้อมูลนั้นทำได้ภายในเวลาประมาณ 2 วินาที และเนื่องจากการกำหนดให้วงจรหลับเป็นเวลาประมาณ 10 นาทีหรือ 600 วินาทีนั้นทำให้อัตราส่วนการตื่นและหลับจะมีค่าประมาณ 1:300 ซึ่งหากใช้อัตราการลดลงของแบตเตอรีจากกรณีของตัวอย่างเส้นประหยาบสีดำแล้วจะได้อายุการใช้งานโดยประมาณเท่ากับ 30 วันไม่ต้องมีการชาร์จประจุเพิ่ม

4.3.2 การส่งข้อมูล

การทดลองการส่งข้อมูลผ่านตัวรับส่งวิทยุนั้นผู้วิจับพบว่าระบบสามารถรับส่งข้อมูลได้เป็นปกติซึ่งเป็นไปตามหลักการพื้นฐาน โดยในงานวิจัยนี้ระยะทางระหว่างตัวส่งและตัวรับวิทยุห่างกันไม่มาก ไม่เกิน 10 เมตร จึงสามารถรับส่งสัญญาณได้ดี เมื่อผู้วิจัยได้ทดลองประสิทธิภาพของตัวรับส่งวิทยุรุ่นที่ใช้นี้ด้วยการนำไปใช้ในการส่งสัญญาณใกล้บ่อนากุ้งที่มีขนาดบ่อยาวประมาณ 40 เมตร ไม่มีแนววัตถุบดบังแนวสายตาระหว่างตัวรับและตัวส่งวิทยุ พบว่าการส่งข้อมูลบางครั้งอาจจะมีช่วงที่ข้อมูลขาดหายไปได้บ้างขึ้นอยู่กับหลายปัจจัย โดยเฉพาะคุณสมบัติของตัวรับส่งวิทยุ, ระยะทาง และคุณภาพของสายอากาศที่ใช้

สำหรับในกรณีของการส่งข้อมูลผ่านจีพีอาร์แอสนั้นพบว่าระบบทำงานได้ตามต้องการ อย่างไรก็ตามผู้วิจัยพบว่าบางครั้งมอดูลจีพีอาร์แอส/จีเอสเอ็มจะถูกจากตัดสัญญาณการเชื่อมต่อและไม่สามารถเชื่อมต่อได้อีกจนกว่าผู้ใช้จะกดปุ่มรีเซตบนมอดูลด้วยมือ เมื่อทดลองการรีเซตด้วยการส่งสัญญาณไปทางขารีเซตของมอดูลนั้นไม่ส่งผลใด ๆ ทั้งนี้ปัญหาประการหนึ่งของมอดูลจีพีอาร์แอส/จีเอสเอ็มคือการใช้พลังงาน ซึ่งการทำงานผ่านจีพีอาร์แอส/จีเอสเอ็มนั้นต้องใช้พลังงานมากพอสมควร ทำให้ระบบไม่สามารถทำงานได้ด้วยแบตเตอรีเพียงลำพัง จะต้องมีระบบการชาร์จประจุหรือแหล่งจ่ายไฟจากภายนอกเข้ามาช่วยในการทำงาน ดังนั้นผู้ที่สนใจอาจจะพัฒนาระบบการส่งข้อมูลระยะไกลนี้ให้เป็น WiFi หรืออาจจะใช้บลูธูท (Bluttooth) ซึ่งมาตรฐานรุ่นใหม่มีอัตราการใช้พลังงานต่ำและส่งสัญญาณได้เร็วและไกลขึ้นกว่าเดิม

อีกประการหนึ่งคือคลังโปรแกรมสำหรับการใช้งานนั้นมีข้อจำกัดอยู่บ้าง โดย ฟังก์ชันที่มีให้นั้นเป็นการทำงานขั้นพื้นฐานของบริการจีพีอาร์แอส ปัญหาที่ผู้วิจัยพบคือการจัดการข้อมูลที่ตอบกลับจากเครื่องคอมพิวเตอร์ให้บริการนั้นไม่มีฟังก์ชันให้จัดการโดยตรง ผู้วิจัยหรือผู้ใช้คลังโปรแกรมจะต้องเขียนโปรแกรมเพื่อจัดการข้อมูลที่ส่งกลับมาด้วยตนเอง และข้อมูลที่ส่งกลับมาส่วนใหญ่จะเป็นการจัดการสายอักขระแทบทั้งสิ้น ทำให้เสียเวลาไปกับการพัฒนาฟังก์ชันส่วนนี้พอสมควร

4.3.3 การอ่านข้อมูลพยากรณ์อากาศ

การอ่านข้อมูลพยากรณ์อากาศจากแหล่งข้อมูลต่าง ๆ นั้นมีปัญหาหลักสำคัญคือการสกัดข้อมูลจากผู้ให้บริการ โดยทั่วไปหากผู้ให้บริการมีบริการข้อมูลเป็นเอพีไอแล้วก็จะเป็นการอำนวยความสะดวกให้กับผู้ใช้ได้ในระดับหนึ่ง อย่างไรก็ตามข้อมูลที่บริการผ่านเอพีไอเหล่านี้บางส่วนยังไม่ได้มีการออกแบบข้อมูลให้เป็นมาตรฐานไปในทางเดียวกัน ทำให้การพัฒนาโปรแกรมเพื่อสกัดข้อมูลต้องทำหลายรอบ ในกรณีของข้อมูลจากกรมอุตุนิยมวิทยานั้นสามารถสกัดข้อมูลได้สองแบบ แบบแรกคือการสกัดจากเว็บปกติซึ่งเป็นเทคนิคที่ใช้ในการศึกษาครั้งนี้ ซึ่งผลการทดลองแสดงให้เห็นว่าสามารถทำได้และได้ผลดี แต่หากหน่วยงานที่บริการข้อมูลนี้มีการเปลี่ยนแปลงเว็บไซต์ ข้อมูลแท็กภายในเอกสารเว็บมีการเปลี่ยนแปลงแล้วก็จะทำให้ระบบทำงานผิดทันที ดังนั้นทางเหลือกอื่นที่อาจจะเหมาะสมกว่าในระยะยาวคือการใช้บริการข้อมูลผ่านเอพีไอ ซึ่งผู้ใช้สามารถเลือกรูปแบบข้อมูลที่จะรับได้ตามความต้องการ