



รายงาน

เรื่อง นาฬิกาดิจิตอล

จัดทำโดย

นางสาวกฤติมา พัฒน์ชัย

นายธนต์ ตันอึ้ง

นายณัฐชนน เกื่อนถ้ำ

ห้อง E31

เสนอ

อาจารย์ปณวีร์ อติศัยศักดิ์ดา

รายงานนี้เป็นส่วนหนึ่งของวิชา Microprocessor Practice

ภาคเรียนที่ 2 ปีการศึกษา 2565

โรงเรียนเตรียมวิศวกรรมศาสตร์ ไทย-เยอรมัน



## คำนำ

รายงานฉบับนี้ เป็นส่วนหนึ่งของชิ้นงานในวิชา Microprocessor Practice การจัดทำรายงานและชิ้นงานนี้ขึ้นมา มีวัตถุประสงค์เพื่อพัฒนาและต่อยอดนาฬิกาดิจิตอลรุ่นก่อนให้มีประสิทธิภาพมากขึ้น ด้วยรุ่นก่อนมีปัญหาในเรื่องโครงสร้างการออกแบบที่ซับซ้อน การใช้ไมโครคอนโทรลเลอร์เกินความจำเป็น ความสว่างไม่เพียงพอและการสูญเสียพลังงานออกไปในรูปแบบความร้อน โดยรุ่นใหม่นี้ได้ทำการเปลี่ยนโครงสร้างใหม่ทั้งหมด ซึ่งสามารถปรับระดับความสว่างได้ ทำให้การสูญเสียลดลง เปลี่ยนวิธีการแสดงผลโดยการใช้ไอซีเลื่อนบิตหรือรีจิสเตอร์แทนที่การแสดงผลด้วยการกระพริบของตัวเลขอย่างรวดเร็ว

แต่รุ่นใหม่นี้ก็ยังมีปัญหาที่เกิดขึ้นอยู่ นั่นก็คือ โมดูลนาฬิกา หรือ Real Time Clock Module ในบางเวลาที่เปิดใช้งานใหม่ๆ ตัว NodeMCU เองไม่สามารถอ่านค่าจากตัวโมดูลนาฬิกาได้จึงจำเป็นต้องถอดตัวโมดูลนาฬิกาออกและใส่เข้าไปใหม่ก็จะใช้งานได้ตามปกติ

อันที่จริงแล้วปัญหาที่เกิดขึ้นระหว่างทำชิ้นงานนั้นมีมากเหลือล้น ยกตัวอย่างเช่นการเลือกอุปกรณ์อย่างทรานซิสเตอร์ จะทำอย่างไรให้เกิดความร้อนน้อยที่สุด แต่อย่างไรก็ตามสุดท้ายแล้วก็สามารถทำให้สำเร็จจนได้ การจัดทำรายงานและนาฬิกาเรือนนี้ขึ้นมานั้น ผู้พัฒนาหวังว่าจะเป็นประโยชน์ให้กับผู้ที่สนใจ และขอขอบคุณที่สละเวลาอันมีค่าของท่านที่อ่านมาถึงตรงนี้ ขอขอบคุณครับ

นายณัฐชนน เกื่อนถ้ำ และคณะ




## สารบัญ

เรื่อง	หน้า
คำนำ .....	ก
สารบัญ.....	๗
ที่มาและความสำคัญ.....	1
คุณสมบัติและสรรพคุณ .....	1
วิธีการใช้งาน .....	2
1. การใช้งาน Buck Converter .....	2
1.1 Buck Converter (Main).....	2
1.2 Buck Converter (Sub).....	3
2. การเชื่อมต่อกับอินเตอร์เน็ต .....	3
3. การปรับความสว่างการแสดงผล .....	3
3.1 การกำหนดความสว่างสูงสุดให้กับหน้าจอแสดงผล.....	3
4. การตั้งค่าการเชื่อมต่ออินเตอร์เน็ตและอื่นๆ .....	4
ข้อควรระวัง.....	6
ปัญหาที่อาจเกิดขึ้นได้.....	6
ส่วนประกอบของชิ้นงาน .....	7
1. โดยภาพรวม .....	7
2. อุปกรณ์ในหน้าจอแสดงผล 7-Segments .....	9
3. อุปกรณ์ในวงจรตัดแรงดันไฟเกินกำหนด .....	9
วงจรของชิ้นงาน .....	10
1. (Diagram) ส่วนการ Control และ Power.....	10
2. (Diagram) ส่วนการแสดงผล 7-Segments.....	11
3. (การต่อจริง) ส่วนการ Control และ Power.....	12
4. (การต่อจริง) ส่วนการแสดงผล 7-Segments.....	13
เกี่ยวกับ 74HC595.....	14
การใช้งานไอซีเลื่อนบิต 74HC595 ด้วย shiftOut() .....	15
โค้ดโปรแกรม .....	17

*KMUTNB DigitalClock Project*

***SENTi***  
***Meg***

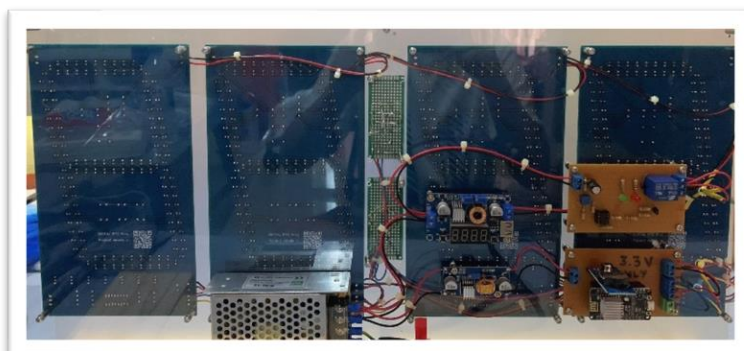
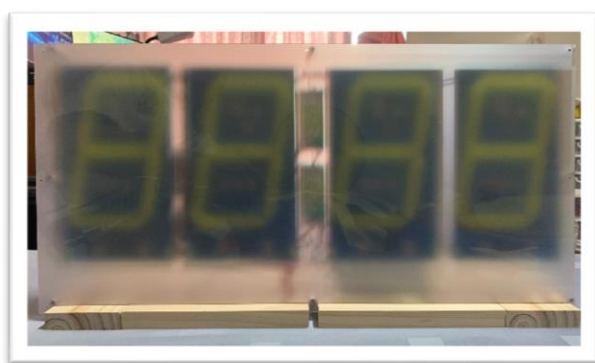
The logo features the text "SENTi" in a bold, italicized sans-serif font, with the "i" having a dot. Below it is the word "Meg" in a similar bold, italicized sans-serif font. The text is partially enclosed by a thin, curved line that starts from the left and ends near the "g". To the right of the text is a stylized starburst or spark graphic, consisting of several sharp, intersecting lines forming a star-like shape.

## ที่มาและความสำคัญ

นาฬิกาดิจิตอลเรือนนี้ได้พัฒนาและต่อยอดมาจากรุ่นก่อนที่มีปัญหาหลายด้าน ด้วยรุ่นก่อนที่มีการออกแบบที่ซับซ้อน เสริมนี้เพิ่มเติมนี้หน่อยจนพันกันมั่วไปหมด อีกทั้งปัญหาจากความร้อนที่เกิดขึ้นทำให้เกิดการสูญเสียพลังงานไฟฟ้าโดยไม่ให้เกิดประโยชน์ และได้รับหัวข้อมานันก็คือการรวมเอาไมโครคอนโทรลเลอร์สองตัวมารวมกันให้เป็นหนึ่ง โดยรุ่นก่อนจะให้ตัว NodeMCU เป็นตัวดึงข้อมูลเวลาจากอินเทอร์เน็ตแล้วส่งให้ตัว Arduino กับโมดูลนาฬิกาแสดงผลออกไปที่ 7-Segments ผ่านระบบการกระพริบที่ละหลักอย่างรวดเร็ว โดยรุ่นใหม่ที่พัฒนามานี้ได้ทำการเปลี่ยนโครงสร้างใหม่ทั้งหมด โดยเปลี่ยนวิธีการแสดงผล โดยการใช้ไอซีเลื่อนบิตหรือรีจิสเตอร์แทน ซึ่งทำให้มีสายสัญญาณที่น้อยลงอย่างมาก ลดภาระของตัวไมโครคอนโทรลเลอร์

## คุณสมบัติและสรรพคุณ

นาฬิกาตั้งโต๊ะหรือนาฬิกาแขวนผนัง (ยกเว้น Smart Watch) นั้นมีปัญหาคือต้องตั้งเวลาด้วยตัวเอง หรือก็คือระบบ Manual ซึ่งอาจจะทำให้เวลาคลาดเคลื่อนได้จากความเป็นจริงหรืออาจจะเก็ยจคร้านในการตั้ง พวกเราจึงได้พัฒนาต่อยอดนาฬิกาให้สามารถตั้งเวลาอัตโนมัติเมื่อมีอินเทอร์เน็ตที่สามารถเชื่อมต่อได้ สามารถเชื่อมต่ออินเทอร์เน็ตใหม่ได้เมื่อสัญญาณขาดหายไปชั่วคราว สามารถปรับความสว่างได้ สามารถตัดระบบเมื่อแรงดันไฟเกินกำหนดซึ่งเกิดจากการปรับความสว่างที่มากเกินไป สามารถแสดงอุณหภูมิห้องได้



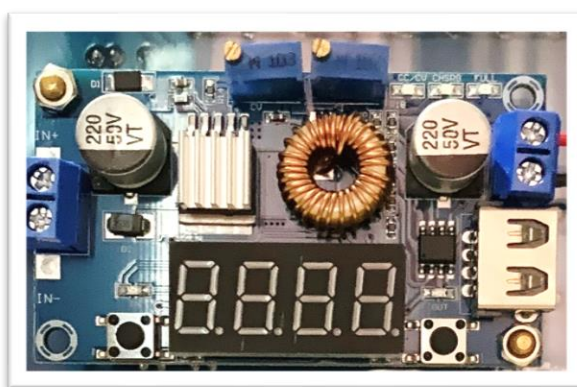
## วิธีการใช้งาน

### 1. การใช้งาน Buck Converter

Buck Converter หรือ Step down เป็นวงจรอิเล็กทรอนิกส์ที่สามารถทำแรงดันไฟตรงให้ที่เข้ามาลดลงได้ เพื่อให้ค่าแรงดันไฟฟ้ามีความเหมาะสมสำหรับการนำไปใช้ในงานอื่นๆ โดยที่นาฬิกาดิจิตอลเรื่อนนี้ได้นำวงจรนี้มาใช้ด้วยกัน 2 ตัวคือ

1. Buck Converter (Main)
2. Buck Converter (Sub)

#### 1.1 Buck Converter (Main)



##### 1.1.1 การปรับและควบคุมแรงดันไฟฟ้ากระแสตรง

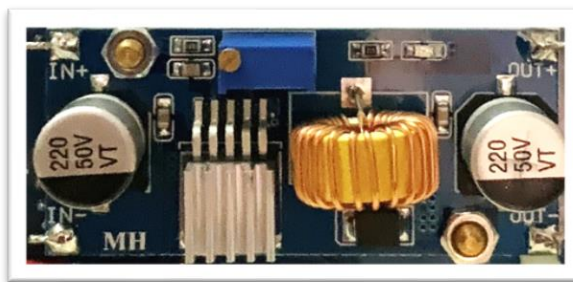
วงจร Step down วงจรนี้ทำหน้าที่จ่ายกำลังหลักให้กับหน้าจอแสดงผลโดยสามารถปรับแรงดันได้โดยการปรับทริมพอร์ทด้านซ้ายมือสุด โดยจะมีสัญลักษณ์ CV หรือ (Control Voltage) กำกับที่ตัวทริมพอร์ต ส่วนทริมพอร์ตด้านขวาสุดจะเป็นการปรับกระแสสูงสุด โดยจะมีสัญลักษณ์ CC หรือ (Control Current) กำกับที่ตัวทริมพอร์ตอยู่

##### 1.1.2 การตั้งค่าหน้าจอแสดงผล

วงจร Step down วงจรนี้มีหน้าจอแสดงผลอยู่และมีปุ่มข้างซ้ายและขวาของจอแสดงผล ปุ่มซ้ายเป็นการปิดหน้าจอแสดงผล ส่วนปุ่มขวาจะเป็นการเลือกโหมดการแสดงผลโดยจะมีอยู่ 4 โหมดดังนี้คือ 1. แรงดันอินพุต 2. แรงดันเอาต์พุต 3. กระแสเอาต์พุต 4. กำลังเอาต์พุต หรือจะดูที่สัญลักษณ์ V (Volt) A (Amp) และ P (Power) ก็ได้ โดยจะมีไฟแสดงสถานะเป็นสีแดงด้านบนปุ่มกดทั้งสอง ถ้าสว่างทางด้านปุ่มซ้ายมือหมายถึงแสดงผลทางด้านอินพุต ถ้าสว่างทางขวามือหมายถึงแสดงผลทางเอาต์พุต

## วิธีการใช้งาน (ต่อ)

### 1.2 Buck Converter (Sub)



#### 1.2.1 การปรับและควบคุมแรงดันไฟฟ้ากระแสตรง

วงจร Step down วงจรนี้ทำหน้าที่จ่ายกำลังให้กับตัวไมโครคอนโทรลเลอร์และไอซีควบคุมต่างๆในซึ่งเป็นส่วนหนึ่งของวงจรแสดงผลโดยสามารถปรับแรงดันได้โดยการปรับทริมพอร์ตสีน้ำเงินตรงกลาง *โดยจะไม่แนะนำให้ทำการปรับแรงดันไฟฟ้าในวงจรนี้* เพราะตัวไมโครคอนโทรลเลอร์เองอาจได้รับความเสียหายได้ โดยวงจรนี้จะทำการคงที่แรงดันไว้ประมาณ 3.28 Volt

## 2. การเชื่อมต่อกับอินเทอร์เน็ต

กระจายไวไฟผ่านมือถือหรือจะเป็นเราเตอร์ที่สามารถเข้าถึงอินเทอร์เน็ตได้ ตัว NodeMCU สามารถเชื่อมต่อสัญญาณใหม่ได้ตลอดเวลาโดยไม่ต้องกดปุ่มใดๆ แต่จำเป็นต้องตั้งค่าชื่อและรหัสผ่านของไวไฟที่ต้องการให้เชื่อมต่อด้วย (จะกล่าวถึงในส่วนถัดๆไป)

## 3. การปรับความสว่างการแสดงผล

การปรับความสว่างการแสดงผลสามารถปรับได้โดยการเพิ่มหรือลดแรงดันไฟตรงที่จ่ายให้กับหน้าจอแสดงผลหลัก โดยการปรับทริมพอร์ตด้านซ้ายมือสุดที่วงจร Buck Converter (Main) โดยจะมีสัญลักษณ์ CV หรือ (Control Voltage) กำกับที่ตัวทริมพอร์ตเอง (สามารถดูเพิ่มเติมได้จากหน้าที่ 2 หัวข้อการปรับและควบคุมแรงดันไฟฟ้ากระแสตรง)

### 3.1 การกำหนดความสว่างสูงสุดให้กับหน้าจอแสดงผล

การกำหนดความสว่างหรือแรงดันสูงสุดให้กับหน้าจอแสดงผลเพื่อป้องกันการเกิดอคริภัยที่อาจเกิดขึ้นได้จากความร้อนได้ที่ตัววงจรสร้างขึ้นเพราะมีแรงดันอินพุตที่สูงเกินไปจากการปรับความสว่างหน้าจอของตัวเอง (วิธีการตั้งค่าอยู่หน้าถัดๆไป)



## วิธีการใช้งาน (ต่อ)



วิธีการตั้งค่าแรงดันสูงสุดมี 3 ขั้นตอน คือ

1. ปรับแรงดันสูงสุดตามที่ต้องการที่ตัว Buck Converter (Main) โดยดูทางด้านแรงดันเอาต์พุต
2. ปรับทริคพอร์ตสีน้ำเงินทางซ้ายมือที่อยู่ในบอร์ดในภาพด้านบนจนกว่าไฟสถานะสีแดงจะสว่างขึ้น หรือถ้าสว่างอยู่แล้วให้ปรับจนกว่าไฟจะดับลงแล้วค่อยๆปรับขึ้นใหม่จนไฟสถานะติดอีกครั้ง
3. ลดแรงดันเอาต์พุตที่ตัว Buck Converter (Main) ที่ตั้งไว้ตอนแรก ให้อยู่ในสถานะปกติ

เมื่อวงจรอยู่ในสถานะปกติจะมีไฟสถานะสีเขียวติดอยู่แสดงว่าวงจรสามารถทำงานได้ ส่วนไฟสถานะสีแดงหมายถึงการตัดการจ่ายไฟหลักให้กับวงจรแสดงผล

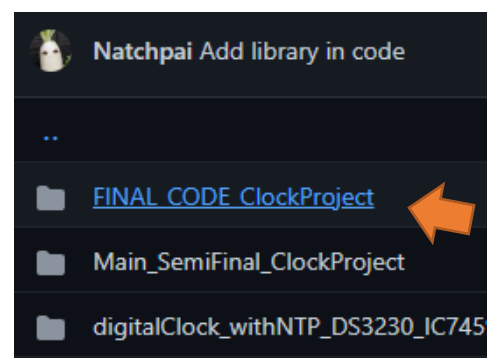
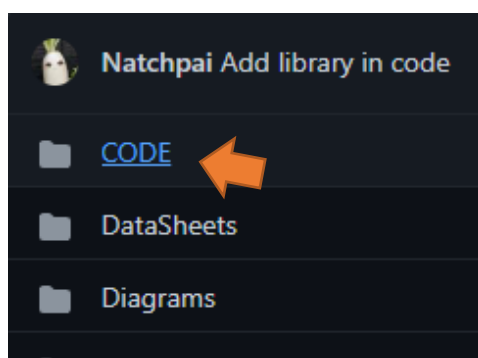
## 4. การตั้งค่าการเชื่อมต่ออินเทอร์เน็ตและอื่นๆ

การตั้งค่าการเชื่อมต่ออินเทอร์เน็ตหรือการเปลี่ยนชื่อและรหัสไวไฟหรือการตั้งค่าต่างๆนั้นจำเป็นอย่างมากที่จะต้องอัปเดตโปรแกรมใหม่ให้กับตัวไมโครคอนโทรลเลอร์ มีวิธีดังนี้

1. สแกน QR CODE ด้านหลังแผงวงจร หรือเข้าลิงก์

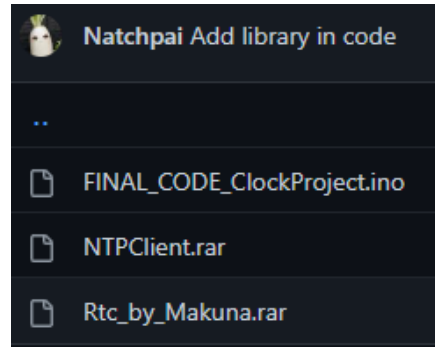
<https://github.com/Natchpai/proDigitalClockE31> (ไม่ใช่ proDigitalClockE31)

2. ไปที่โฟลเดอร์ชื่อ CODE / FINAL\_CODE\_ClockProject ตามรูปด้านล่าง (แต่ถ้าสแกน QR CODE ให้ไปที่โฟลเดอร์ชื่อ DigitalClockProject ก่อน)



## วิธีการใช้งาน (ต่อ)

3. FINAL\_CODE\_ClockProject.ion เป็นโค้ดที่จำเป็นต้องลง Library บางตัว สามารถดูเพิ่มเติมได้ในโค้ด ส่วนในโฟลเดอร์นี้จะมีมาให้ 2 ตัว สามารถลงได้เลย



4. สามารถเปลี่ยน SSID และ password แล้วอัปโหลดใหม่ได้เลย ส่วน pullData\_hour คือการตั้งเวลาทุกๆ ชั่วโมง เมื่อมีการเชื่อมต่ออินเทอร์เน็ต ในที่นี้คือ ทุกๆ 6 ชม.

```

25  const char* ssid = "NatchPai";
26  const char* password = "powerpay4";
27
28  int16_t pullData_hour = 6;

```

5. สามารถเปลี่ยนเวลาการเปลี่ยนโหมดระหว่างแสดงเวลากับอุณหภูมิได้  
ในเงื่อนไข mode == 1 คือแสดงเวลา ส่วน mode == 2 คือแสดงอุณหภูมิ

```

231  if (mode == 1) {
232      // 40 Seconds
233      if(countSquare == (40) ) {
234          countSquare = 0;
235          digitalWrite(DOTpin, 0);

```

```

240  else if(mode == 2) {
241      // 8 Seconds
242      if(countSquare == (8) ) {
243          countSquare = 0;
244          digitalWrite(DOTpin, 0);

```



## ข้อควรระวัง

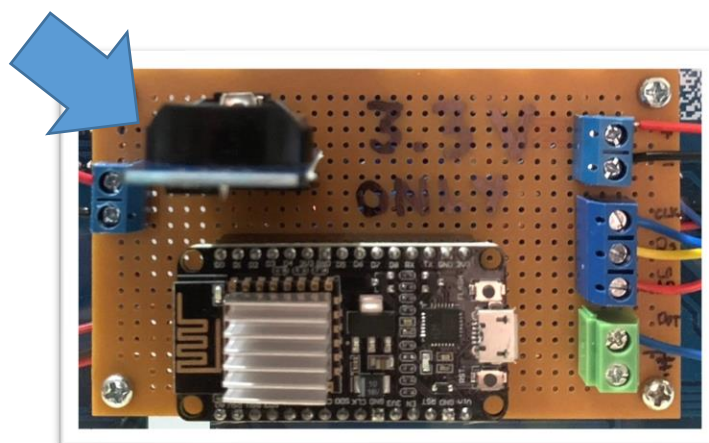
1. ไม่แนะนำให้ปรับแรงดันที่วงจร Buck Converter (Sub) เพราะอาจจะทำให้ไมโครคอนโทรลเลอร์เสียหายได้ ซึ่งวงจรนี้จะคงแรงดันไว้ที่ 3.28 Volt โดยประมาณ
2. ไม่แนะนำให้ปรับ CC หรือ Control Current ที่วงจร Buck Converter (Main)
3. ไม่แนะนำให้ตั้ง CC หรือ CV ที่วงจร Buck Converter (Main) ผ่านการกดปุ่ม
4. อย่าปรับความสว่างหรือแรงดันมากจนเกินเพราะอาจเกิดอัคคีภัยได้
5. ระวังแผ่นอะคริลิกแตก

## ปัญหาที่อาจเกิดขึ้นได้

ปัญหาโมดูลนาฬิกา หรือ Real Time Clock Module ในบางเวลาที่เปิดใช้งานใหม่ๆ ตัว NodeMCU เองไม่สามารถอ่านค่าจากตัวโมดูลนาฬิกาได้ ทำให้เกิดการแสดงผลเวลาที่ผิดเป็นเวลา 00:00 และ 00°C

มีวิธีแก้ด้วยกัน 2 วิธีคือ

1. ถอดปลั๊กนาฬิกาออกจากเต้าแล้วเสียบเข้าไปใหม่อีกที หรือ
2. ดึงโมดูลนาฬิกาออกและใส่เข้าไปใหม่ขณะที่นาฬิกากำลังทำงานอยู่ (ใส่ให้ถูกด้านด้วย)



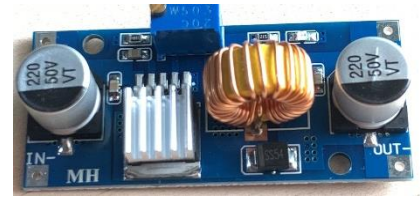
## ส่วนประกอบของชิ้นงาน

### 1. โดยภาพรวม

1. หน้าจอแสดงผล 7-Segments x 4 แผ่น



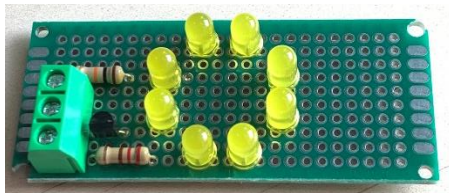
5. Buck Converter ขนาด 2 แอมป์



6. NodeMCU ESP8266 V2



2. หน้าจอแสดงผลแบบ Dot x 2 แผ่น



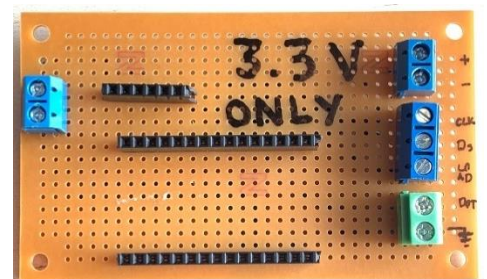
7. DS3231 (RTC Module)



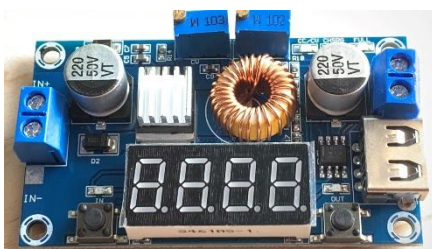
3. ฐานไม้ แผ่นอะคริลิกใส และแผ่นอะคริลิกขุ่น



8. ฐานใส่ NodeMCU, RTC Module



4. Buck Converter ขนาด 5 แอมป์



9. วงจรตัดแรงดันไฟเกินกำหนด





## ส่วนประกอบของชิ้นงาน (ต่อ)

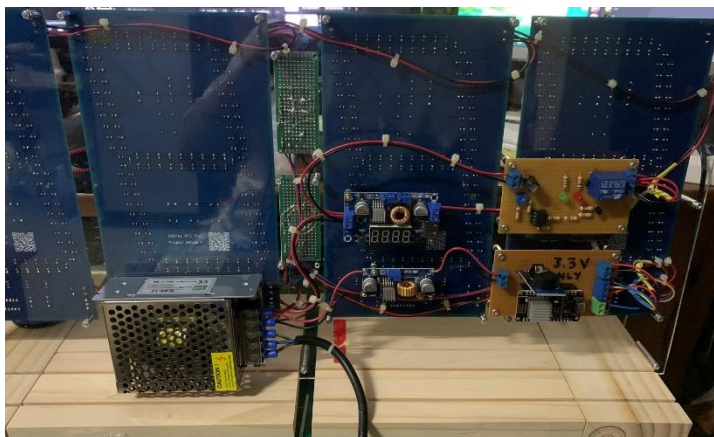
10. Power Supply ขนาด 220VAC แปลงเป็น 12VDC-5A.



ภาพรวมด้านหน้า



ภาพรวมด้านหลัง



## ส่วนประกอบของชิ้นงาน (ต่อ)

### 2. อุปกรณ์ในหน้าจอแสดงผล 7-Segments

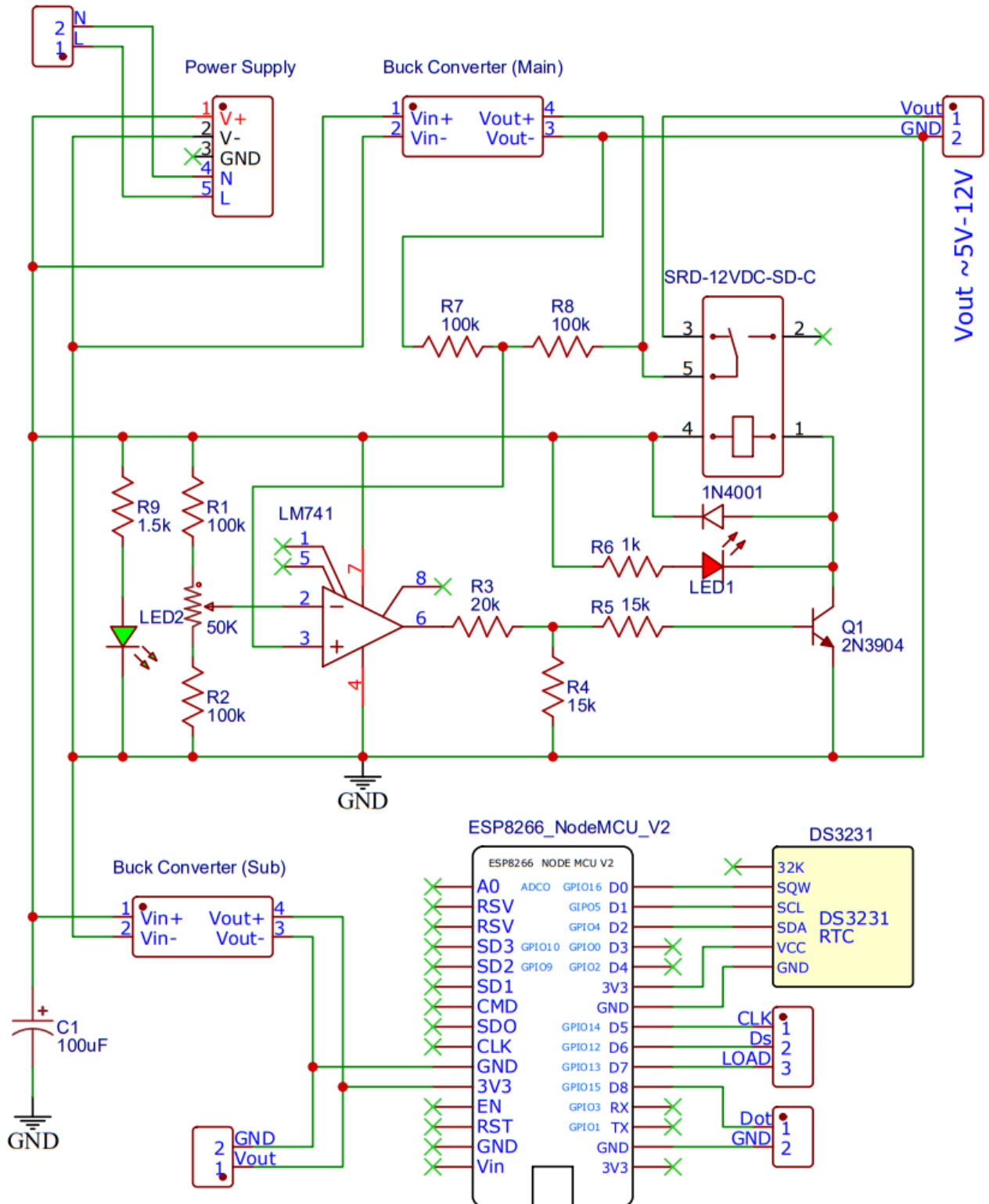
1. แผ่น PCB x 1 แผ่น
2. หลอด LED สีเหลืองขนาด 5mm. x 140 ดวง
3. ตัวต้านทาน 220  $\Omega$  x 14 ตัว
4. ตัวต้านทานขนาด 10 K $\Omega$  x 7 ตัว
5. ทรานซิสเตอร์เบอร์ 2N3904 x 7 ตัว
6. ไอซีเบอร์ 74HC595 พร้อม Socket 16 pin x 1 ตัว
7. ตัวเก็บประจุชนิดเซรามิก 0.1 $\mu$ F x 1 ตัว
8. Screw Terminal ขนาด 5mm. 2 pin x 2 ตัว
9. Screw Terminal ขนาด 3.5mm. 3 pin x 2 ตัว

### 3. อุปกรณ์ในวงจรตัดแรงดันไฟเกินกำหนด

1. รีเลย์ไฟตรงขนาด 12 โวลต์ เบอร์ SRD-12VDC-SL-C x 1 ตัว
2. ไดโอดเบอร์ 1N4001 x 1 ตัว
3. หลอด LED 5mm. สีแดงและสีเขียว อย่างละ 1 ดวง
4. ตัวต้านทานขนาด 15 K $\Omega$  x 2 ตัว
5. ตัวต้านทานขนาด 10 K $\Omega$  x 2 ตัว
6. ตัวต้านทานขนาด 100 K $\Omega$  x 4 ตัว
7. ตัวต้านทานขนาด 1 K $\Omega$  x 1 ตัว
8. ตัวต้านทานขนาด 1.5 K $\Omega$  x 1 ตัว
9. ทริมพอร์ต 1 รอบ ขนาด 50 K $\Omega$  x 1 ตัว
10. ไอซีเบอร์ LM741 พร้อม Socket 8 pin x 1 ตัว
11. ทรานซิสเตอร์เบอร์ 2N3904 x 1 ตัว
12. ตัวเก็บประจุชนิดอิเล็กโทรไลต์ขนาด 100 $\mu$ F 50V x 1 ตัว
13. Screw Terminal ขนาด 5mm. 2 pin x 1 ตัว
14. Screw Terminal ขนาด 5mm. 3 pin x 1 ตัว
15. แผ่นวงจรสำเร็จ IC3 x 1 แผ่น

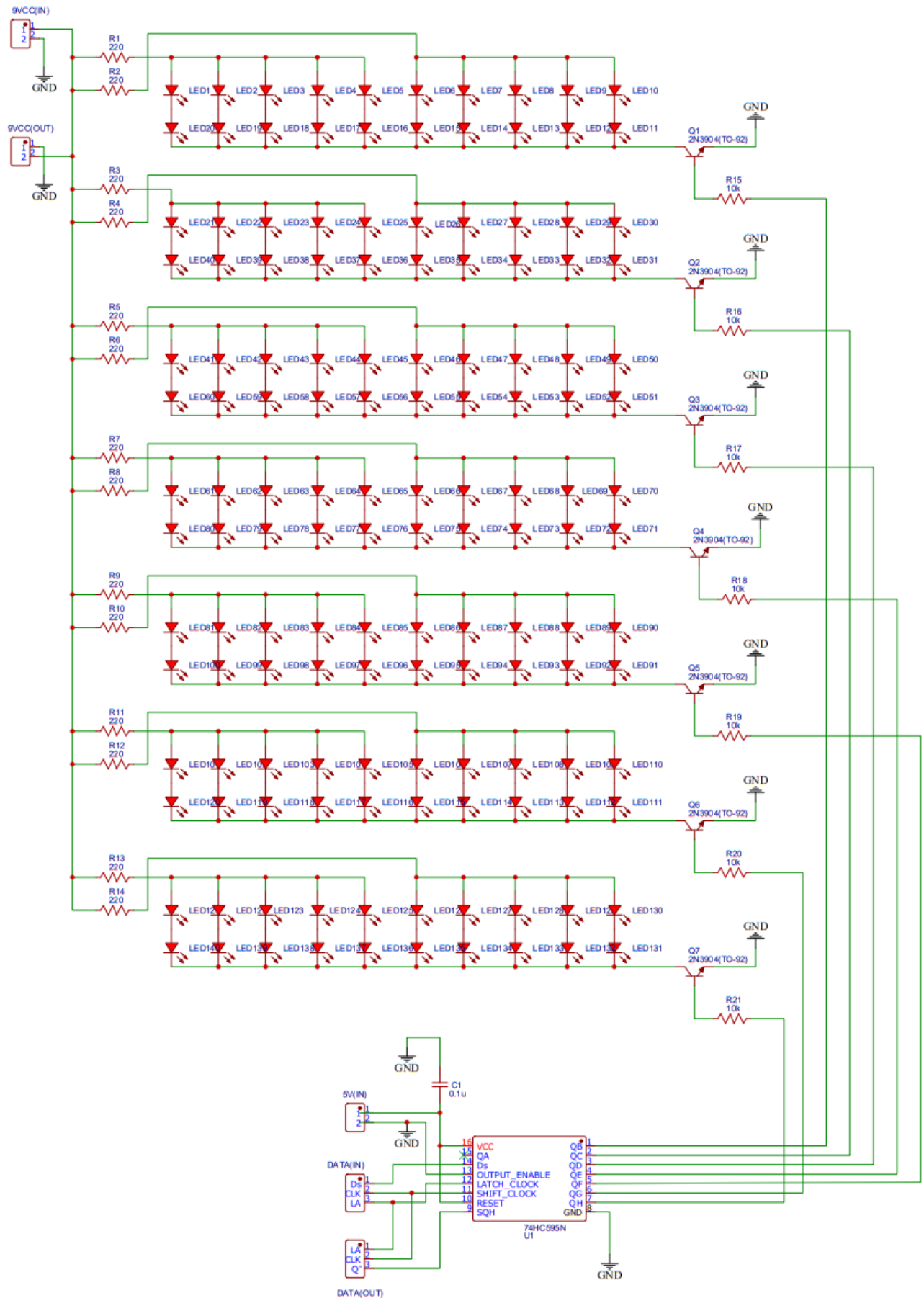
## วงจรของชิ้นงาน

### 1. (Diagram) ส่วนการ Control และ Power



## วงจรของชิ้นงาน (ต่อ)

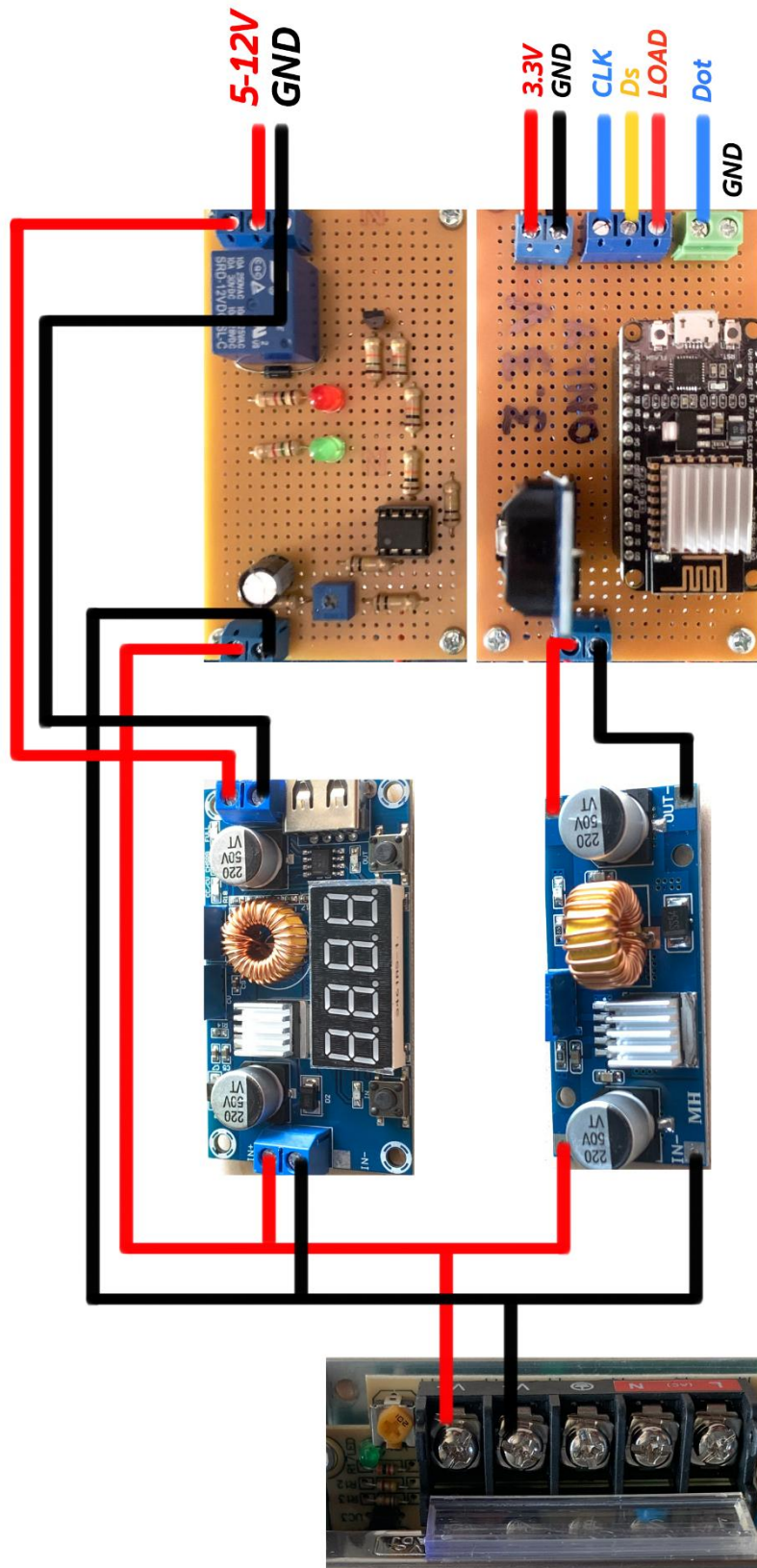
### 2. (Diagram) ส่วนการแสดงผล 7-Segments





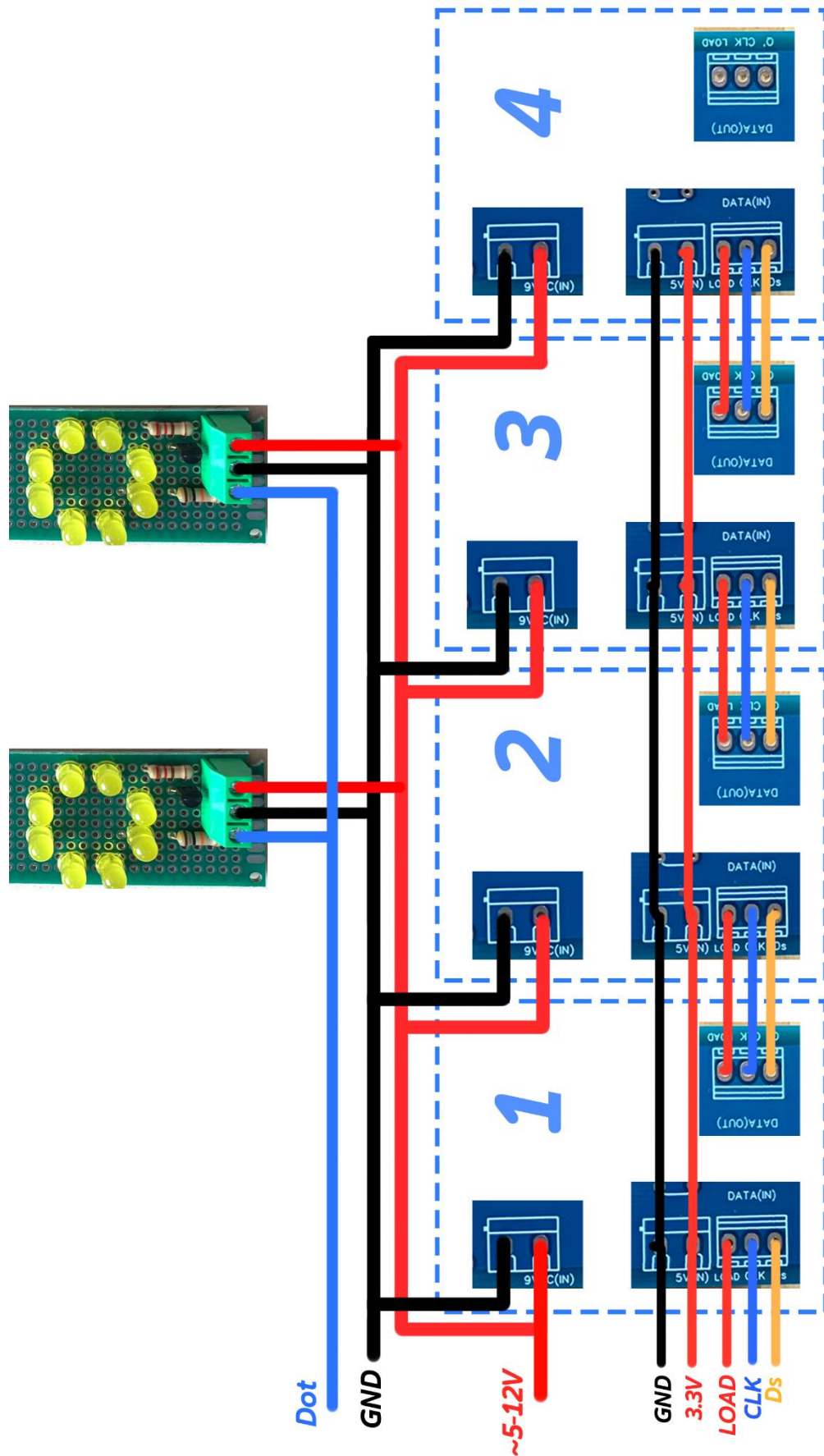
วิจารณ์ของชิ้นงาน (ต่อ)

### 3. (การต่อจริง) ส่วนการ Control และ Power



## วงจรของชิ้นงาน (ต่อ)

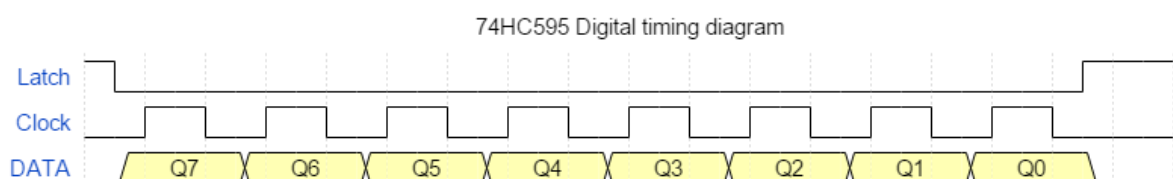
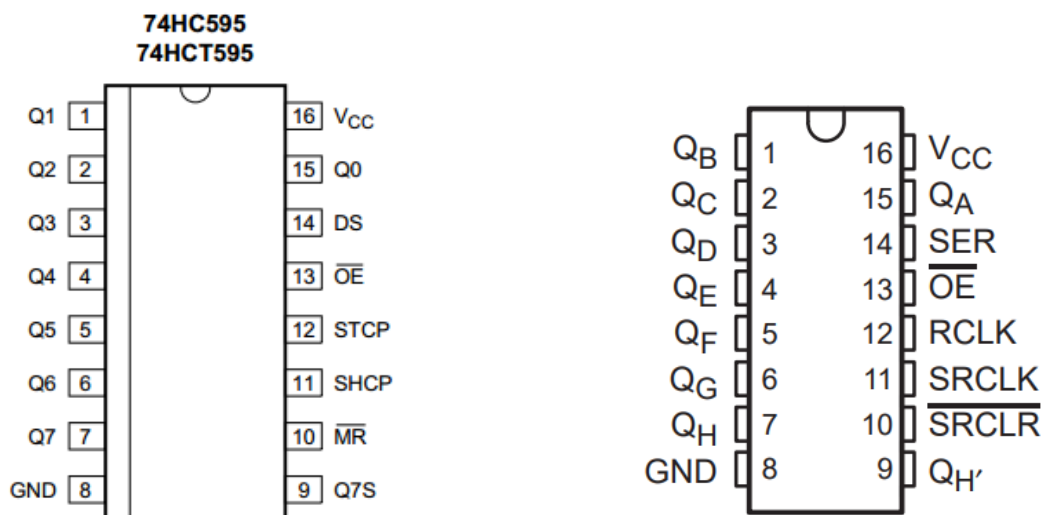
### 4. (การต่อจริง) ส่วนการแสดงผล 7-Segments



## เกี่ยวกับ 74HC595

ไอซี 74HC595 เป็นไอซีเลื่อนบิต เมื่อมีการป้อนข้อมูลเข้าไปใหม่ บิตจะถูกเลื่อนไปตามข้อมูลที่ป้อน มีขาทั้งหมด 16 ขา เป็นขาเอาต์พุตที่ควบคุมได้ทั้งหมด 8 ขา ตั้งแต่ Q0 Q1 Q2 ... Q7 มีขา 3 ขาสำหรับการควบคุมขา Q0 - Q7 ไอซีชนิดนี้จะมาช่วยในเรื่องการขยายขาส่งข้อมูลดิจิตอลให้มากขึ้น

1. ขา ST\_CP หรือเทียบได้กับการ Latch เป็นขาควบคุมจังหวะการส่งข้อมูลออก เมื่อขานี้มีสถานะเป็น HIGH จะมีการโหลดข้อมูลออกไปที่ขา Q0 Q1 Q2 ... Q7 ตามข้อมูลที่ส่งเข้ามา
2. ขา SH\_CP เทียบได้กับขา Clock เป็นขาที่จะต้องป้อนสัญญาณพัลส์เข้าไปเพื่อควบคุมการรับข้อมูลเข้าไอซี โดยการป้อนสัญญาณจะต้องสัมพันธ์กับการป้อนข้อมูลในแต่ละบิต
3. ขา DS เป็นขาสำหรับป้อนข้อมูลเข้าไปที่ละบิตตามจังหวะสัญญาณนาฬิกา
4. ขา Q7S หรือ QH' เป็นขาข้อมูลออก สามารถนำไปต่อกับขา DS ของตัวถัดๆไปเพื่อขยายความสามารถในการส่งข้อมูลได้
5. ขา MR หรือขา Reset ทำงานในสถานะ LOW
6. ขา OE หรือ Output Enable Input ทำงานในสถานะ LOW



\* Latch = ST\_CP, Clock = SH\_CP, DATA = DS

## เกี่ยวกับ 74HC595

### การใช้งานไอซีเลื่อนบิต 74HC595 ด้วย shiftOut()

ฟังก์ชัน shiftOut เป็นฟังก์ชันส่งข้อมูลแบบซิงโครนัส โดยสามารถกำหนดขา data และ clock เข้าไปในฟังก์ชันได้ โดยในที่นี้ 74HC595 นั้นสามารถรับข้อมูลได้มากที่สุด 8 บิตต่อหนึ่งตัว แต่สามารถนำไอซีแต่ละตัวมาต่ออนุกรมกันเพื่อขยายความสามารถในการเก็บข้อมูลให้มากขึ้นได้

`shiftOut(dataPin, clockPin, bitOrder, value);`

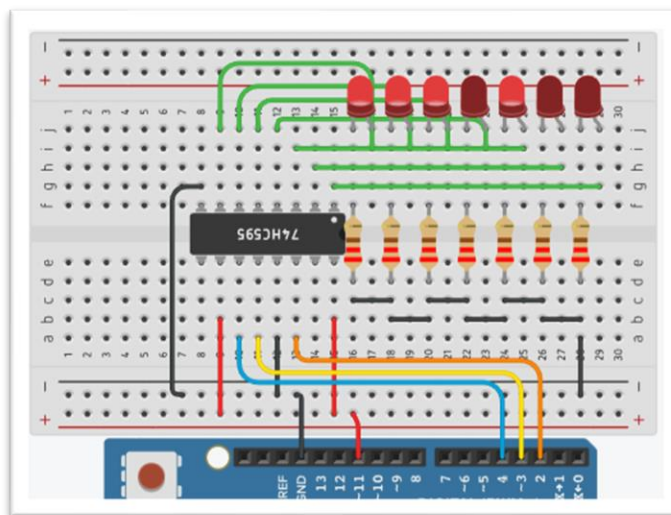
**dataPin** คือ ขาที่ต่อใช้งานกับขา Ds ของ 74HC595

**clockPin** คือ ขาที่ต่อใช้งานกับขา SH\_CP ของ 74HC595

**bitOrder** คือ การตั้งค่าการเรียงลำดับของข้อมูลที่จะส่งออกว่าจะส่งบิตความสำคัญสูงหรือบิตความสำคัญต่ำออกก่อน MSBFIRST or LSBFIRST

**value** คือข้อมูลจำนวน 8 บิตที่จะส่งออก สามารถใช้ค่าจากตัวแปร byte หรือ uint8\_t

1. ตัวอย่างการใช้งานฟังก์ชัน shiftOut กับ 74HC595 แบบ 8 บิต



```
digitalWrite(3, 0);
shiftOut(2, 4, LSBFIRST, 0b0010111);
digitalWrite(3, 1);
digitalWrite(3, 0);
```



## โค้ดโปรแกรม

```

1  #include <ESP8266WiFi.h>
2  #include <NTPClient.h>
3  #include <WiFiUdp.h>
4  #include <Arduino.h>
5
6
7  // Shift Register SN74HC595N 8-bit SIPO
8  #define SRCLK D5
9  #define SER_DATA D6
10 #define LATCH D7
11 #define DOTpin D8
12 uint8_t digits[10] = {126, 48, 109, 121, 51, 91, 95, 112, 127, 123}; // 0 -> 9 from binary
13 uint8_t celsiusUnit[2] = {0b01100011, 0b01001110};
14 // 0b 0 X X X X X X X
15 //      a b c d e f g
16
17
18 // DS3230
19 #include <Wire.h>
20 #include <RtcDS3231.h>
21 RtcDS3231<TwoWire> Rtc(Wire);
22 // D1 -> SCL, D2 -> SDA
23 #define SquareWave_pin D0 // D0 -> SQW
24 uint8_t SquareWave;
25
26
27 const char* ssid = "NatchPai";
28 const char* password = "powerpay4";
29
30 int16_t pullData_hour = 48;
31 bool connection;
32 bool onePullQuota = true;
33
34 int timezone = 7 * 3600;
35 int dst = 0;
36 WiFiUDP ntp;
37 NTPClient timeClient(ntp, "europe.pool.ntp.org", timezone);
38
39 unsigned long previousPullDataTimes;
40 unsigned long previousTimes;
41 unsigned long previousTimes2;
42 unsigned long currentTimes;
43 unsigned long oldLoadTimes;
44 unsigned long newLoadTimes;
45
46 void setup() {
47     WiFi.mode(WIFI_STA);
48     WiFi.begin(ssid, password);
49     // Serial.begin(9600);
50
51     //Set WIFI
52     timeClient.begin();
53     WiFi.setAutoReconnect(true);
54     WiFi.persistent(true);
55

```



## โค้ดโปรแกรม

```

56 //Set DS3231
57 Rtc.Begin();
58 Rtc.SetSquareWavePin(DS3231SquareWavePin_ModeClock); //Sets pin mode
59 Rtc.SetSquareWavePinClockFrequency(DS3231SquareWaveClock_1Hz); //Sets frequency
60 checkStateRTC();
61 pinMode(SquareWave_pin, INPUT);
62
63 // Set 74595
64 pinMode(LATCH, OUTPUT);
65 pinMode(SRCLK, OUTPUT);
66 pinMode(SER_DATA, OUTPUT);
67 pinMode(DOTpin, OUTPUT);
68
69 TestStart();
70 }
71
72 void TestStart() {
73   for(int i=9;i>=0;i--) {
74     digitalWrite(LATCH, 0);
75     shiftOut(SER_DATA, SRCLK, LSBFIRST, digits[i]);
76     digitalWrite(DOTpin, !digitalRead(DOTpin));
77     digitalWrite(LATCH, 1);
78     digitalWrite(LATCH, 0);
79     delay(100);
80   }
81   for(int i=0;i<=4;i++) {
82     digitalWrite(LATCH, 0);
83     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
84     digitalWrite(LATCH, 1);
85     digitalWrite(LATCH, 0);
86     delay(100);
87   }
88
89   LatchData(0b01000000, 0b00000000, 0b00000000, 0b00000000, 126, 126); delay(100);
90   LatchData(0b01000000, 0b01000000, 0b00000000, 0b00000000, 126, 126); delay(100);
91   LatchData(0b01000000, 0b01000000, 0b01000000, 0b00000000, 126, 126); delay(100);
92   LatchData(0b01000000, 0b01000000, 0b01000000, 0b01000000, 126, 126); delay(100);
93   LatchData(0b01000000, 0b01000000, 0b01000000, 0b01100000, 126, 126); delay(100);
94   LatchData(0b01000000, 0b01000000, 0b01000000, 0b01110000, 126, 126); delay(100);
95   LatchData(0b01000000, 0b01000000, 0b01000000, 0b01111000, 126, 126); delay(100);
96   LatchData(0b01000000, 0b01000000, 0b01001000, 0b01111000, 126, 126); delay(100);
97   LatchData(0b01000000, 0b01001000, 0b01001000, 0b01111000, 126, 126); delay(100);
98   LatchData(0b01001000, 0b01001000, 0b01001000, 0b01111000, 126, 126); delay(100);
99   LatchData(0b01001100, 0b01001000, 0b01001000, 0b01111000, 126, 126); delay(100);
100  LatchData(0b01001110, 0b01001000, 0b01001000, 0b01111000, 126, 126); delay(200);
101  display_SET(); delay(300);
102  ResetDisplay(); delay(50);
103 }

```

## โค้ดโปรแกรม

```

105 void display_SET() {
106     digitalWrite(LATCH, 0);
107     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b01100011);
108     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00001111);
109     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b01001111);
110     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b01011011);
111     digitalWrite(LATCH, 1);
112     digitalWrite(LATCH, 0);
113     delay(1000);
114 }
115
116
117 // Wifi ESP8266 > below
118 void checkStatusWifi() {
119     if(WiFi.status() == WL_CONNECTED) {
120         autoPullData();
121     }
122     else{
123         onePullQuota = true;
124     }
125 }
126
127
128 // Real Time Clock > below
129 void checkStateRTC() {
130     Rtc.Enable32kHzPin(false);
131 }
132
133 void autoPullData() {
134     if (onePullQuota) {
135         updateDate();
136         onePullQuota = false;
137         previousPullDataTimes = 0;
138         display_SET();
139     }
140
141     currentTimes = millis();
142     if( (currentTimes - previousPullDataTimes) > (pullData_hour * 3600000)) {
143         previousPullDataTimes = currentTimes;
144         updateDate();
145     }
146
147     if (currentTimes < previousTimes) {previousPullDataTimes = 0;}
148 }
149

```



## โค้ดโปรแกรม

```

150 void updateDate() {
151     timeClient.update();
152     time_t epochTime = timeClient.getEpochTime();
153     struct tm *ptm = gmtime ((time_t *) &epochTime);
154
155     uint16_t year = ptm -> tm_year + 1900;
156     uint8_t month = ptm -> tm_mon + 1;
157     uint8_t dayMonth = ptm -> tm_mday;
158     uint8_t hour = timeClient.getHours();
159     uint8_t min = timeClient.getMinutes();
160     uint8_t sec = timeClient.getSeconds() + 1; // 1 is Fix delay
161
162     Rtc.SetDateTime(RtcDateTime(year, month, dayMonth, hour, min, sec));
163 }
164
165
166 // Shift Register > below
167 uint8_t rawSec, rawMinute, rawHour;
168 uint8_t second_First, second_End;
169 uint8_t minute_First, minute_End;
170 uint8_t hour_First, hour_End;
171
172 uint8_t partitionFirstDigit(uint8_t data) {
173     if( int(data) >= 10) return data / 10;
174     else return 0;
175 }
176
177 uint8_t partitionEndDigit(uint8_t data) {
178     return int(data) % 10;
179 }
180
181 void analyzeData(RtcDateTime now) {
182     rawSec = now.Second();
183     rawMinute = now.Minute();
184     rawHour = now.Hour();
185     // partitionDigit below
186     second_First = partitionFirstDigit(rawSec);
187     second_End = partitionEndDigit(rawSec);
188     minute_First = partitionFirstDigit(rawMinute);
189     minute_End = partitionEndDigit(rawMinute);
190     hour_First = partitionFirstDigit(rawHour);
191     hour_End = partitionEndDigit(rawHour);
192 }
193
194 // SHIFT DATA Below
195
196
197 // LatchData(hour_first, hour_last, minute_first, minute_last, second_first, second_last);
198 void LatchData(uint8_t Q5, uint8_t Q4, uint8_t Q3, uint8_t Q2, uint8_t Q1, uint8_t Q0) {
199     digitalWrite(LATCH, 0);
200     shiftOut(SER_DATA, SRCLK, LSBFIRST, Q0);
201     shiftOut(SER_DATA, SRCLK, LSBFIRST, Q1);
202     shiftOut(SER_DATA, SRCLK, LSBFIRST, Q2);
203     shiftOut(SER_DATA, SRCLK, LSBFIRST, Q3);
204     shiftOut(SER_DATA, SRCLK, LSBFIRST, Q4);
205     shiftOut(SER_DATA, SRCLK, LSBFIRST, Q5);
206     digitalWrite(LATCH, 1);
207     digitalWrite(LATCH, 0);
208 }

```

## โค้ดโปรแกรม

```

209
210 void ResetDisplay() {
211     digitalWrite(DOTpin, 0);
212     digitalWrite(LATCH, 0);
213     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
214     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
215     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
216     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
217     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
218     shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
219     digitalWrite(LATCH, 1);
220     digitalWrite(LATCH, 0);
221 }
222
223
224 // Auto Mode Set Below
225 uint8_t mode = 1;
226 uint8_t oldStat;
227 unsigned int countSquare;
228
229 void setMode() {
230     SquareWave = digitalRead(SquareWave_pin);
231     // 1 Hz SquareWave to 1s.
232     if (SquareWave == 1) {
233         if(oldStat == 0) {
234             countSquare++;
235             oldStat = 1;
236         }
237     }
238     else if(SquareWave == 0) {
239         oldStat = 0;
240     }
241
242     // 1 Hz SquareWave to 500ms.
243     // if (SquareWave != oldStat) {
244     //     countSquare++;
245     //     oldStat = SquareWave;
246     // }
247
248     if (mode == 1) {
249         // 40 Seconds
250         if(countSquare == (40) ) {
251             countSquare = 0;
252             digitalWrite(DOTpin, 0);
253             mode = 2;
254         }
255     }
256
257     else if(mode == 2) {
258         // 8 Seconds
259         if(countSquare == (8) ) {
260             countSquare = 0;
261             digitalWrite(DOTpin, 0);
262             mode = 1;
263         }
264     }
265 }
266

```

## โค้ดโปรแกรม

```

268 void blinkDot() {
269     if (SquareWave == 1) {
270         digitalWrite(DOTpin, 0);
271     }
272     else {
273         digitalWrite(DOTpin, 1);
274     }
275 }
276
277
278 // MAIN below
279 void loop() {
280     // Load time every 50 ms.
281     // Approximate 20 Frame per second
282     newLoadTimes = millis();
283     if (newLoadTimes - oldLoadTimes >= 50) {
284         oldLoadTimes = newLoadTimes;
285         checkStatusWifi();
286         setMode();
287         if (mode == 1) {
288             blinkDot();
289             RtcDateTime now = Rtc.GetDateTime();
290             analyzeData(now);
291             // LATCH DATA
292             LatchData(digits[hour_First], digits[hour_End], digits[minute_First],
293             digits[minute_End], digits[second_First], digits[second_End]);
294         }
295         else if (mode == 2) {
296             RtcTemperature temp = Rtc.GetTemperature();
297
298             uint8_t temp_F = partitionFirstDigit(temp.AsFloatDegC());
299             uint8_t temp_E = partitionEndDigit(temp.AsFloatDegC());
300             // LATCH DATA
301             digitalWrite(LATCH, 0);
302             shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
303             shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
304             shiftOut(SER_DATA, SRCLK, LSBFIRST, celsiusUnit[1]);
305             shiftOut(SER_DATA, SRCLK, LSBFIRST, celsiusUnit[0]);
306             shiftOut(SER_DATA, SRCLK, LSBFIRST, digits[temp_E]);
307             shiftOut(SER_DATA, SRCLK, LSBFIRST, digits[temp_F]);
308             digitalWrite(LATCH, 1);
309             digitalWrite(LATCH, 0);
310         }
311     }
312     if (newLoadTimes < oldLoadTimes) {oldLoadTimes = 0;}
313 }
314
315
316

```