KMUTNB DigitalClock Project

SENTi Meg

KMUTNB DigitalClock Project

SENTi Meg

DIGITAL CLOCK

DIGITAL REAL TIME CLOCK WITH MEGA 7-SEGMENTS DISPLAY

TABLE OF CONTENTS

01

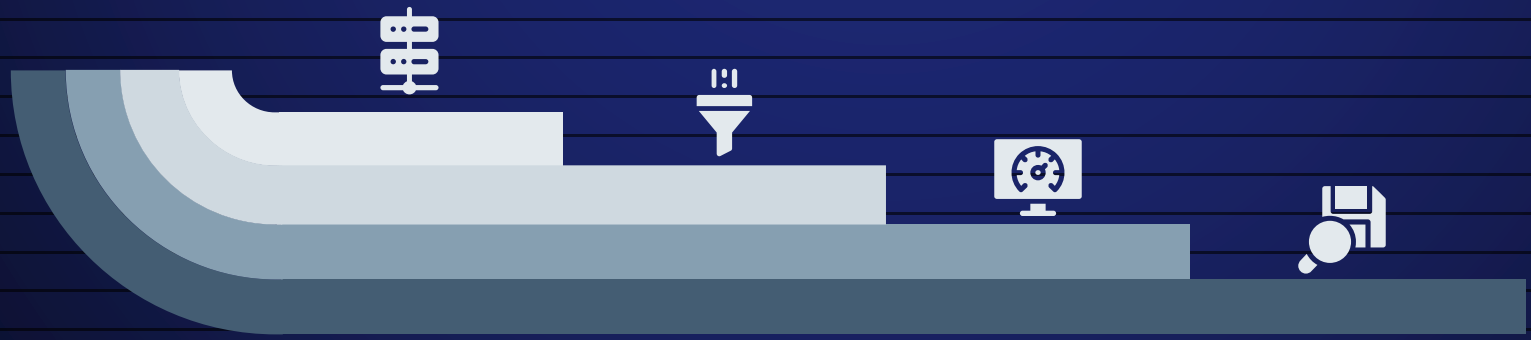
ABOUT TOPIC

เกี่ยวกับหัวข้อเรื่อง ว่าด้วยเรื่องของนาฬิการุ่นก่อน

เกี่ยวกับหัวข้อเรื่อง

นาฬิกาดิจิตอลเรือนนี้ได้พัฒนาและต่อยอดมาจากรุ่นก่อนที่มีปัญหาหลายด้าน ด้วยรุ่นก่อนที่มีการออกแบบที่ซับซ้อน อีกทั้งปัญหาจากความร้อนที่เกิดขึ้นทำให้เกิดการสูญเสียพลังงานไฟฟ้าโดยไม่ให้เกิดประโยชน์ และได้รับหัวข้อมานั้นก็คือการรวมเอาไมโครคอนโทรลเลอร์สองตัวมารวมกันให้เป็นหนึ่ง

โดยรุ่นก่อนจะให้ตัว NodeMCU เป็นตัวดึงข้อมูลเวลาจากอินเตอร์เน็ตแล้วส่งให้ตัว Arduino กับโมดูลนาฬิกาแสดงผลออกไปที่ 7-Segments ผ่านระบบการกระพริบทีละหลักอย่างรวดเร็ว



02

PROBLEM

เกี่ยวกับปัญหาต่างๆของนาฬิการุ่นก่อนที่ยากจะแก้

ปัญหาที่เกิดขึ้น

ความสว่างไม่เพียงพอ

- น่าจะเกิดจากแรงดันไบอัสหลอดไม่พอ หรืออาจจะกระแสไม่พอก็ได้

มีความร้อนเกิดขึ้นสูง

- ยังไม่ทรายสาเหตุที่แน่ชัด

ใช้ไมโครคอนโทรลเลอร์สองตัว

- น่าจะเกิดจากจำนวนขาของ NodeMCU ไม่พอในการสั่ง 7-Segments

การเดินสายไม่เรียบร้อย

- ยังไม่ทรายสาเหตุที่แน่ชัด

03

NEW PRODUCT

สิ่งที่พัฒนาและความสามารถของรุ่นใหม่ที่ขึ้นกว่าเดิมแน่นอน

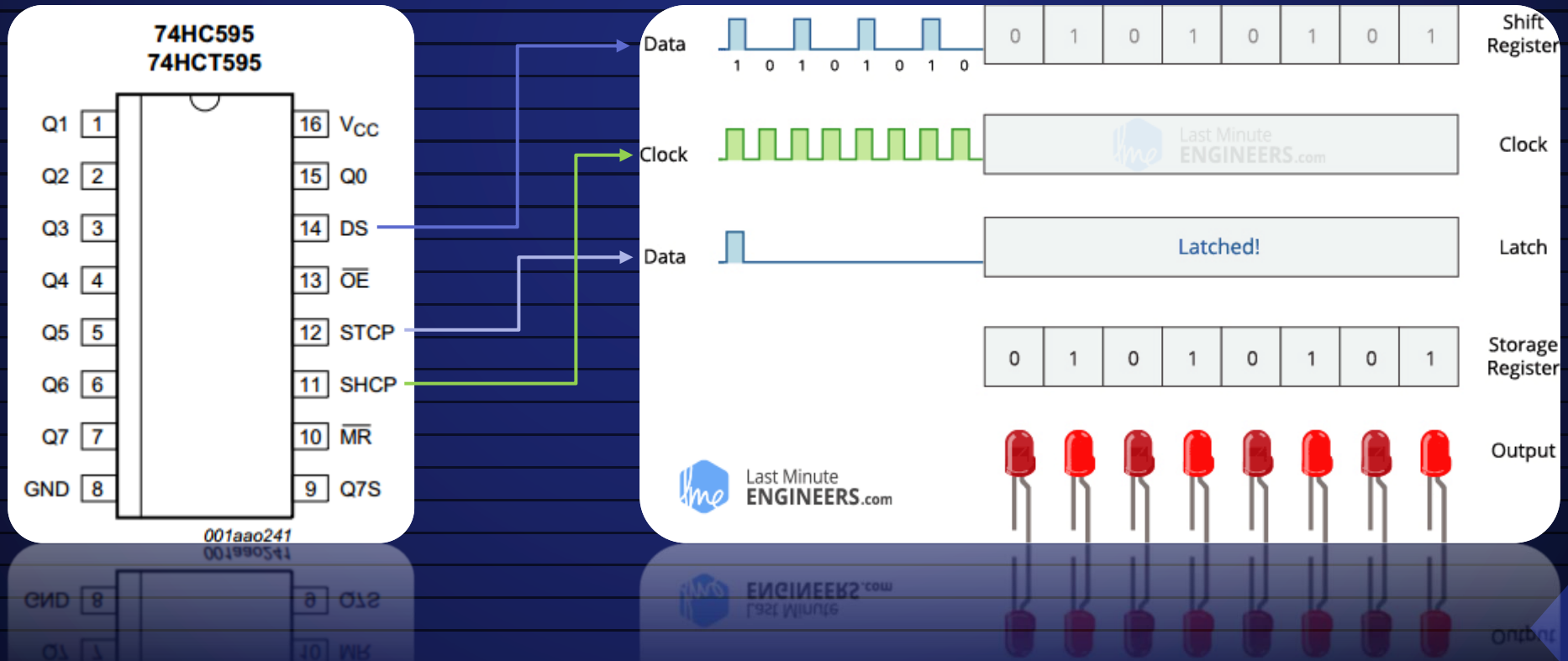
ABOUT PRODUCT

นาฬิกาตั้งโต๊ะหรือนาฬิกาแขวนผนัง นั้นมีปัญหาคือต้องตั้งเวลาด้วยตัวเอง ซึ่งอาจจะทำให้เวลาคลาดเลื่อนได้จากความเป็นจริงหรืออาจจะเกียจคร้านในการตั้ง

พวกเราจึงได้ออกแบบระบบใหม่ทั้งหมด โดยเปลี่ยนจากการแสดงผลด้วยการกระพริบทีละหลักอย่างรวดเร็วเป็นการใช้ไอซีเลื่อนบิต 74HC595 ซึ่งทำให้มีสายสัญญาณที่น้อยกว่ามาก

ไอซีเลื่อนบิต 74HC595

ไอซี 74HC595 เป็นไอซีเลื่อนบิตที่นำข้อมูลเข้าแบบอนุกรม หรือทีละบิตตามจังหวะสัญญาณนาฬิกา นำข้อมูลออกแบบขนานตามสัญญาณจังหวะการส่งข้อมูลออก และมีการนำข้อมูลออกแบบอนุกรมด้วย



ความสามารถในการทำงาน

ตั้งเวลาอัตโนมัติ

- ตั้งเวลาเมื่อเชื่อมต่ออินเตอร์เน็ต

เชื่อมต่ออัตโนมัติ

- ความสามารถ Reconnect

ตัดแรงดันไฟเกิน

- ตัดแรงดันไฟเกินกำหนด

สามารถปรับความสว่างได้

- เพื่อการประหยัดพลังงาน

วัดอุณหภูมิห้อง

- สามารถตั้งเวลาการแสดงผลได้

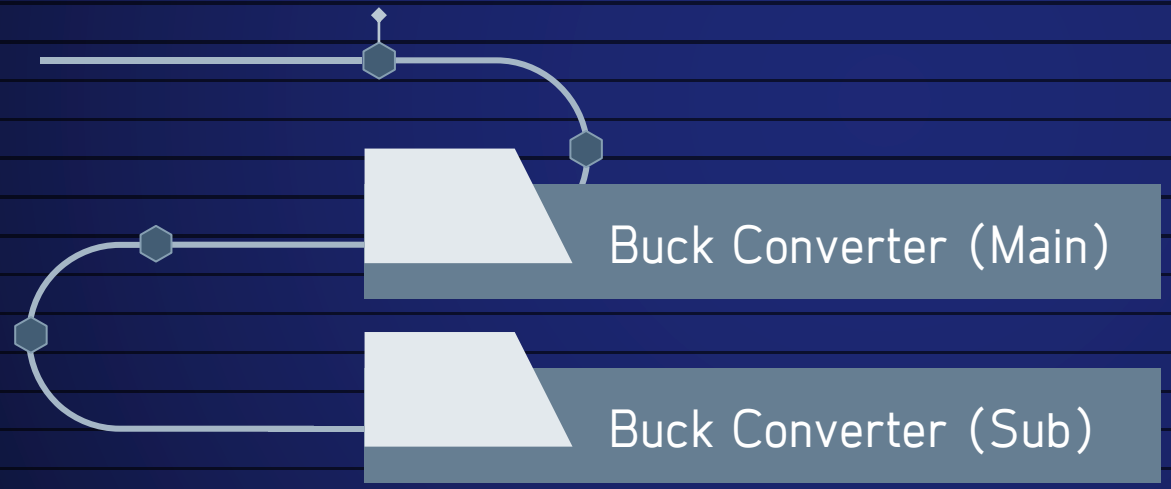
04

USAGE & MANUAL

คู่มือการใช้งานและการแก้ไขปัญหาที่อาจเกิดขึ้นได้

การใช้งาน Buck Converter

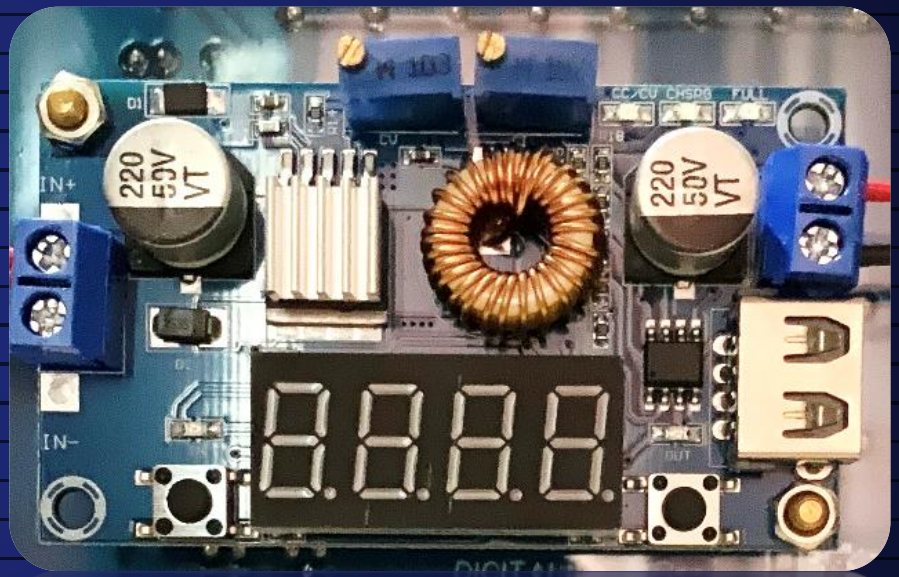
Buck Converter หรือ Step down เป็นวงจรอิเล็กทรอนิกส์ที่สามารถทำแรงดันไฟตรงให้ที่เข้ามาลดลงได้ เพื่อให้ค่าแรงดันไฟฟ้ามีความเหมาะสมสำหรับการนำไปใช้ในงานอื่นๆ โดยที่นาฬิกาดิจิตอลเรือนนี้ได้นำวงจรนี้มาใช้ด้วยกัน 2 ตัวคือ



การใช้งาน Buck Converter (Main)

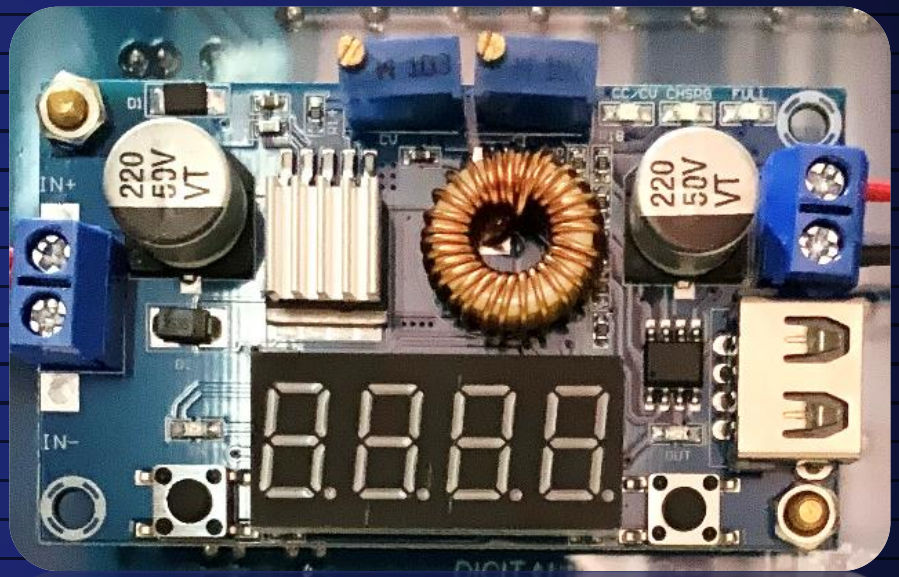
1. การปรับและควบคุมแรงดันไฟฟ้ากระแสตรง

วงจรนี้ทำหน้าที่จ่ายกำลังหลักให้กับหน้าจอแสดงผลโดยสามารถปรับแรงดันได้โดยการปรับทริมพอร์ตด้านซ้ายมือสุด โดยจะมีสัญลักษณ์ CVหรือ (Control Voltage) กำกับที่ตัวทริมพอร์ต



การใช้งาน Buck Converter (Main)

2. การตั้งค่าหน้าจอแสดงผล

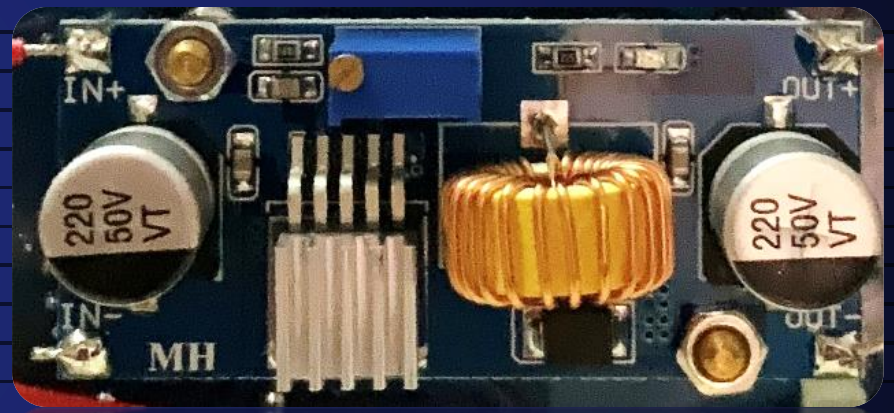


วงจรนี้มีหน้าจอแสดงผลอยู่และมีปุ่มข้างซ้ายและขวาของจอแสดงผล ปุ่มซ้ายเป็นการปิดหน้าจอแสดงผล ส่วนปุ่มขวาจะเป็นการเลือกโหมดการแสดงผลโดยจะมีอยู่ 4 โหมดดังนี้คือ 1. แรงดันอินพุต 2. แรงดันเอาต์พุต 3. กระแสเอาต์พุต 4. กำลังเอาต์พุต โดยจะมีไฟแสดงสถานะเป็นสีแดงด้านบนปุ่มกดทั้งสอง ด้าสว่างทางด้านปุ่มซ้ายมือหมายถึงแสดงผลทางด้านอินพุต ด้าสว่างทางขวามือหมายถึงแสดงผลทางเอาต์พุต

การใช้งาน Buck Converter (Sub)

1. การปรับและควบคุมแรงดันไฟฟ้ากระแสตรง

วงจรนี้ทำหน้าที่จ่ายกำลังให้กับตัวไมโครคอนโทรลเลอร์และไอซีควบคุมต่างๆในซึ่งเป็นส่วนหนึ่งของวงจรแสดงผลโดยสามารถปรับแรงดันได้โดยการปรับทริมพอร์ตสีน้ำเงินตรงกลาง โดยจะไม่แนะนำให้ทำการปรับแรงดันไฟฟ้าในวงจรนี้เพราะตัวไมโครคอนโทรลเลอร์เองอาจได้รับความเสียหายได้ โดยวงจรนี้จะทำการคงที่แรงดันไว้ประมาณ 3.28 Volt



การเชื่อมต่อกับอินเตอร์เน็ต

กระจายไวไฟผ่านมือถือหรือจะเป็นเร้าเตอร์ที่สามารถเข้าถึงอินเตอร์เน็ตได้ ตัว NodeMCU สามารถเชื่อมต่อสัญญาณใหม่ได้ตลอดเวลาโดยไม่ต้องกดปุ่มใดๆ แต่จำเป็นต้องตั้งค่าชื่อและรหัสผ่านของไวไฟที่ต้องการให้เชื่อมต่อด้วย (จะกล่าวถึงในส่วนถัดๆไป)

การปรับความสว่างการแสดงผล

การปรับความสว่างการแสดงผลสามารถปรับได้โดยการเพิ่มหรือลดแรงดันไฟตรงที่จ่ายให้กับหน้าจอแสดงผลหลัก โดยการปรับทริมพอร์ตด้านซ้ายมือสุดที่วงจร Buck Converter (Main) โดยจะมีสัญลักษณ์ CV หรือ (Control Voltage) กำกับที่ตัวทริมพอร์ต

การกำหนดความสว่างสูงสุดให้กับหน้าจอแสดงผล

การกำหนดความสว่างหรือแรงดันสูงสุดให้กับหน้าจอแสดงผลเพื่อป้องกันการเกิดอัคคีภัยที่อาจเกิดขึ้นได้จากความร้อนได้ที่ตัววงจรสร้างขึ้นเพราะมีแรงดันอินพุตที่สูงเกินไปจากการปรับความสว่างหน้าจอ

วิธีการตั้งค่าแรงดันสูงสุดมี 3 ขั้นตอน คือ

1. ปรับแรงดันสูงสุดตามที่ต้องการที่ตัว Buck Converter(Main) โดยดูแรงดันเอาต์พุต

2. ปรับทริมพอร์ตสีน้ำเงินทางซ้ายมือที่อยู่ในบอร์ดในภาพด้านบนจนกว่าไฟสถานะสีแดงจะสว่างขึ้นหรือถ้าสว่างอยู่แล้วให้ปรับจนกว่าจะไฟจะดับลงแล้วค่อยๆ ปรับขึ้นใหม่จนไฟสถานะติดอีกครั้ง

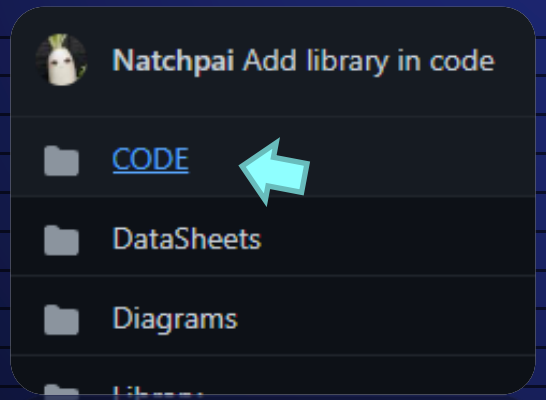
3. ลดแรงดันเอาต์พุตที่ตัว Buck Converter (Main) ที่ตั้งไว้ตอนแรก ให้อยู่ในสถานะปกติ

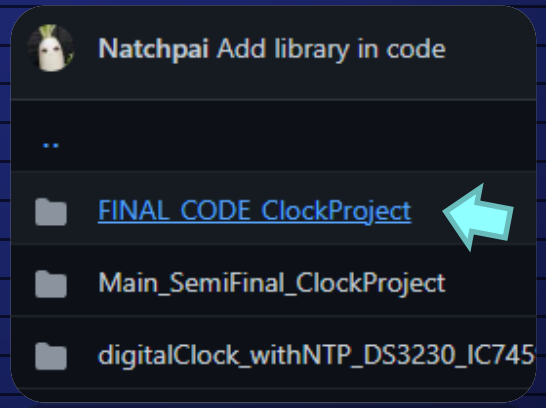
การตั้งค่าการเชื่อมต่ออินเตอร์เน็ตและอื่นๆ

การตั้งค่าการเชื่อมต่ออินเตอร์เน็ตหรือการเปลี่ยนชื่อและรหัสไวไฟหรือการตั้งค่าต่างๆนั้น จำเป็นอย่างมากที่จะต้องอัปโหลดโปรแกรมใหม่ให้กับตัวไมโครคอนโทรลเลอร์ มีวิธีดังนี้

1. สแกน QR CODE ด้านหลังแผงวงจร หรือเข้าลิงก์ https://github.com/Natchpai/proDigialClockE31 (ไม่ใช่ proDigitalClockE31)

2. ไปที่โฟลเดอร์ CODE / FINAL_CODE_ClockProject ตามรูปด้านล่าง แต่ถ้าสแกน QR CODE ให้ไปที่โฟลเดอร์ชื่อ DigitalClockProject ก่อน





การตั้งค่าการเชื่อมต่ออินเตอร์เน็ตและอื่นๆ

3. FINAL_CODE_ClockProject.ion เป็นโค้ดที่จำเป็นต้องลง Library บางตัว สามารถดูเพิ่มเติมได้ในโค้ด ส่วนในโฟลเดอร์นี้จะมีมาให้ 2 ตัว สามารถลงได้เลย

4. สามารถเปลี่ยน SSID และ password แล้วอัปโหลดใหม่ได้เลย ส่วน pullData_hour คือการตั้งเวลาทุกๆกี่ชั่วโมง เมื่อมีการเชื่อมต่ออินเตอร์เน็ต ในที่นี้คือ ทุกๆ 6 ชม.

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

การตั้งค่าการเชื่อมต่ออินเตอร์เน็ตและอื่นๆ

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);
```

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

⚠ ข้อควรระวัง

1. ไม่แนะนำให้ปรับแรงดันที่วงจร Buck Converter (Sub) เพราะอาจจะทำให้วงจรเสียหายได้ 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. ระวังอะคริลิกแตก

⚠ ปัญหาที่อาจเกิดขึ้นได้

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

มีวิธีแก้ด้วยกัน 2 วิธีคือ

1. ถอดปลั๊กนาฬิกาออกจากเต้าแล้วเสียบเข้าไปใหม่อีกที หรือ
2. ดึงโมดูลนาฬิกาออกและใส่เข้าไปใหม่ขณะที่นาฬิกากำลังทำงานอยู่

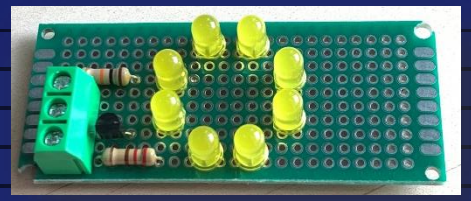
05

EQUIPMENT & DIAGRAM

อุปกรณ์ต่างๆ การต่อวงจร และ Diagram

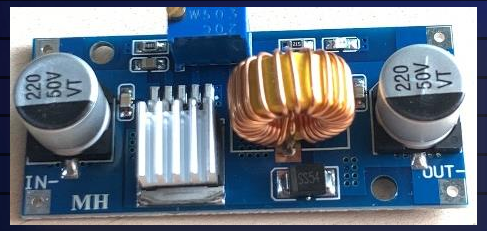
ส่วนประกอบ

ฐานไม้ แผ่นอะคริลิกใส และ
แผ่นอะคริลิกขุ่น



หน้าจอแสดงผลแบบ Dot
x 2 แผ่น

หน้าจอแสดงผล 7-Segments
x 4 แผ่น



Buck Converter ขนาด 2 แอมป์

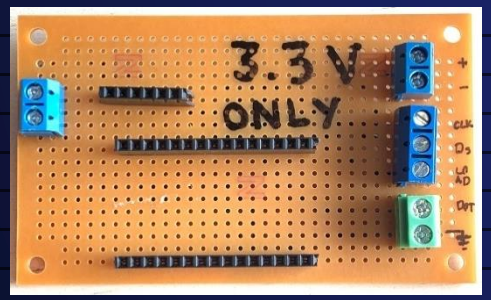
Buck Converter ขนาด 5 แอมป์

ส่วนประกอบ

NodeMCU ESP8266 V2

DS3231 (RTC Module)

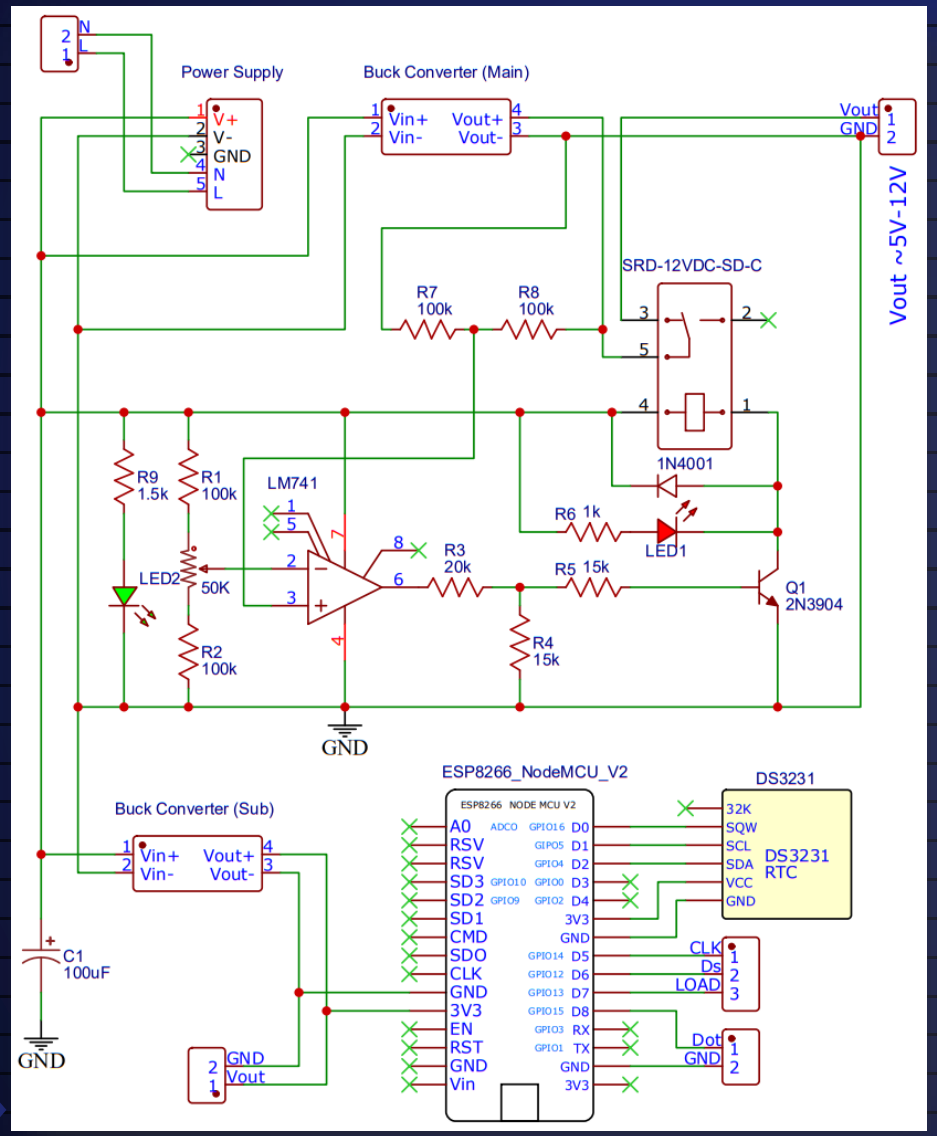
Power Supply ขนาด 220VAC
แปลงเป็น 12VDC-5A

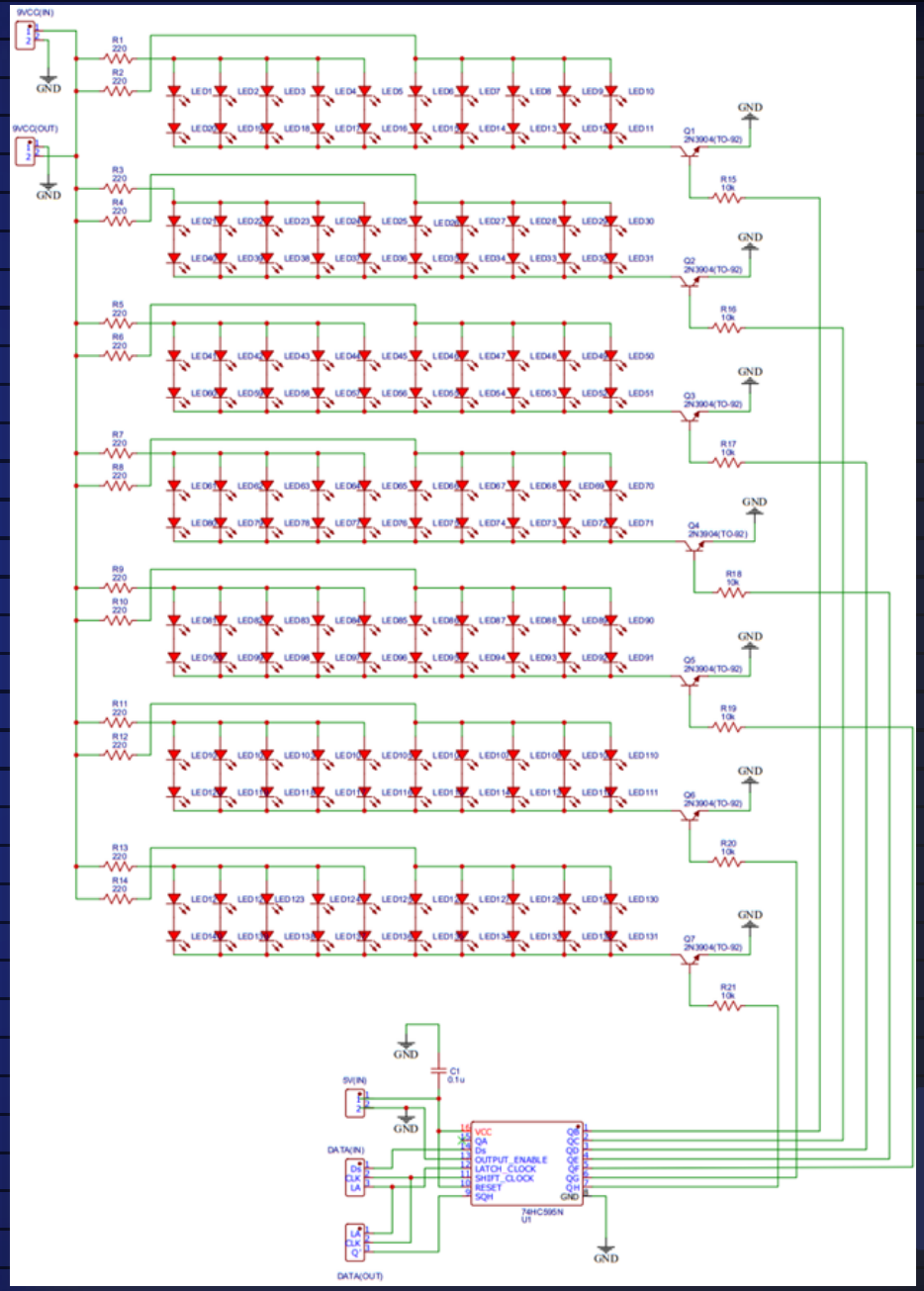


ฐานใส่ NodeMCU, RTC Module

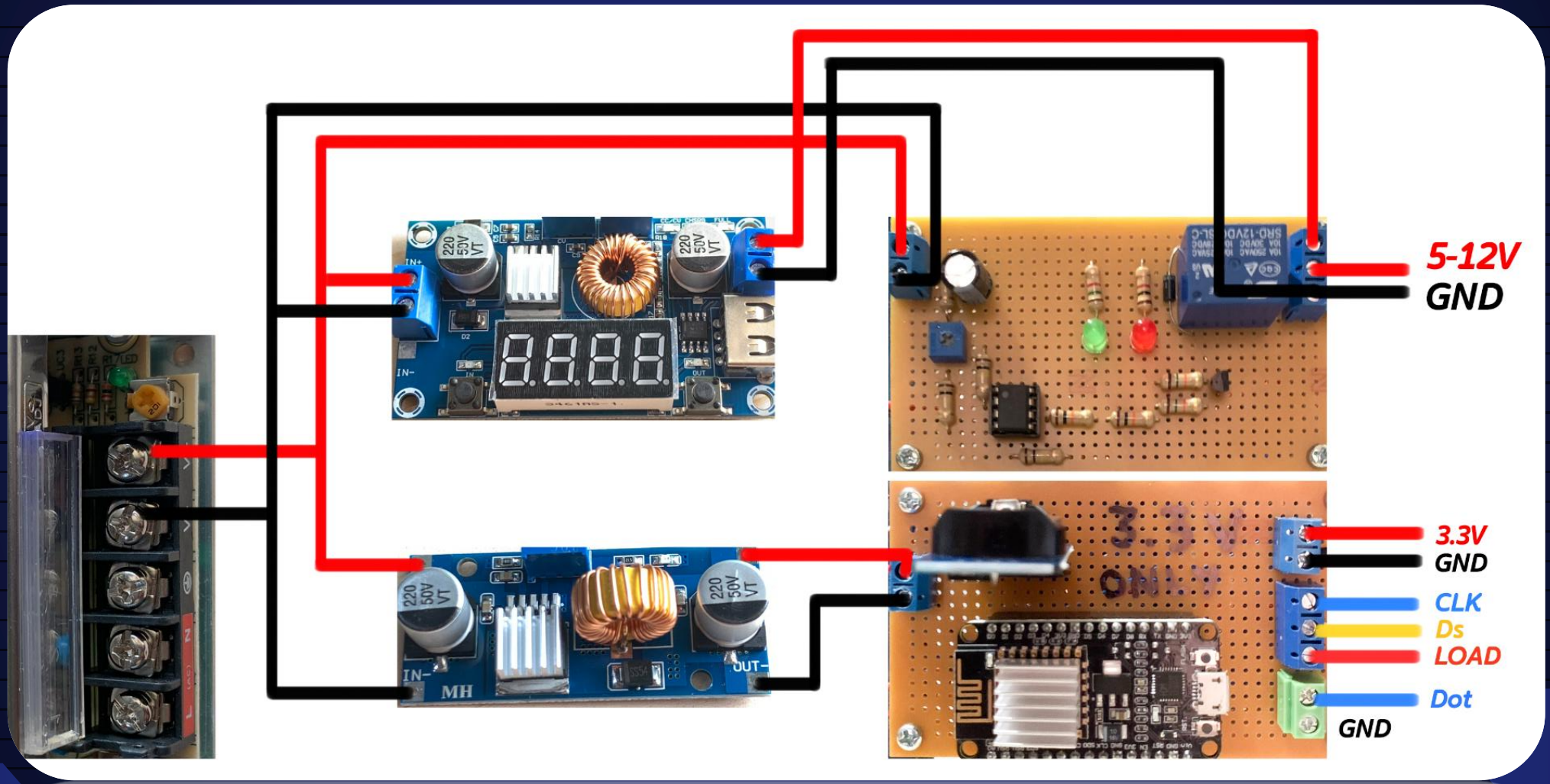
วงจรตัดแรงดันไฟเกินกำหนด

DIAGRAM

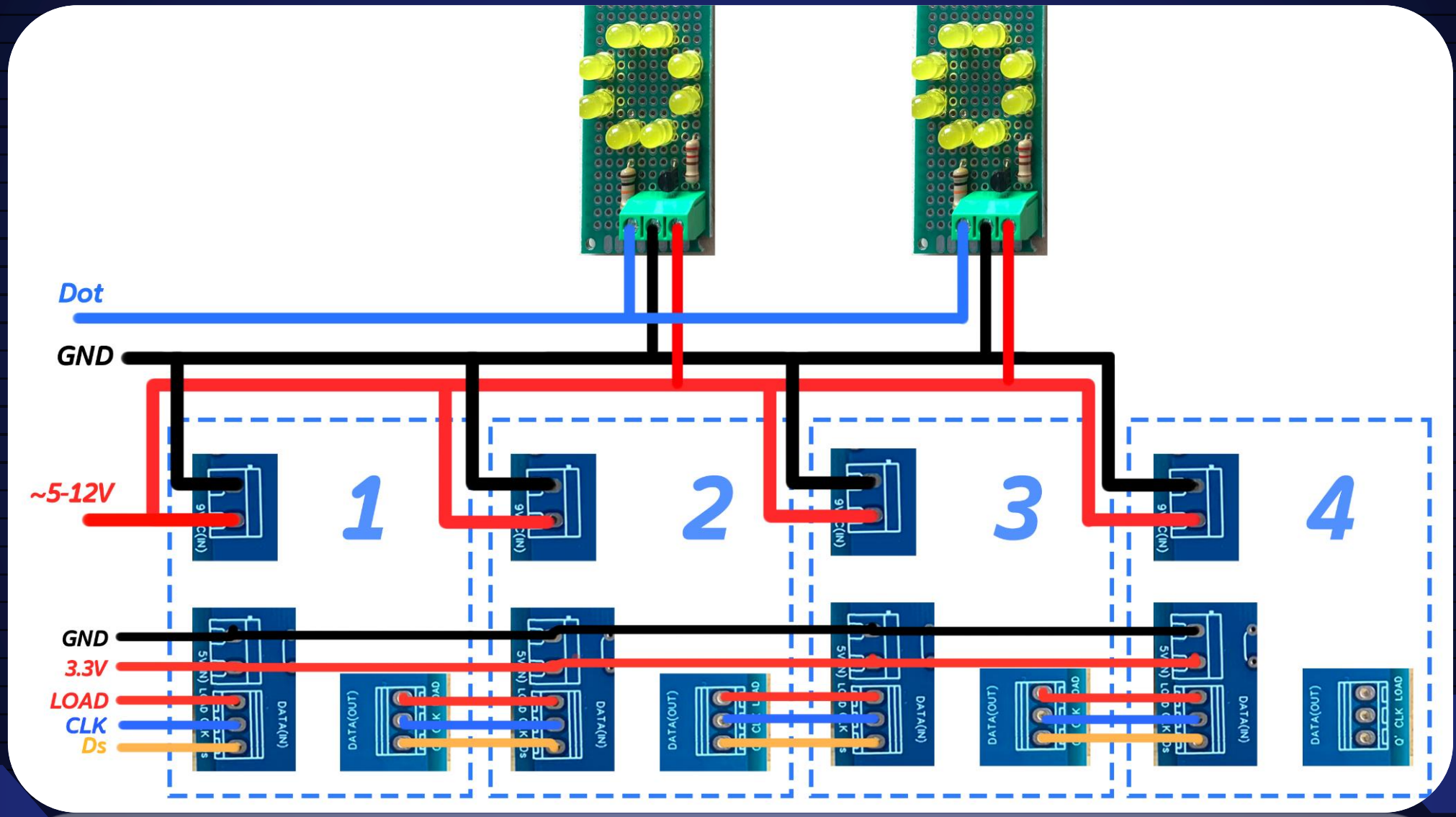

Power Supply
Buck Converter (Main)
Vout
GND
Vout ~5V-12V
SRD-12VDC-SD-C
R7
100k
R8
100k
R9
1.5k
R1
100k
LM741
R6 1k
1N4001
LED1
LED2
50K
R3
20k
R5 15k
Q1
2N3904
R2
100k
R4
15k
GND
Buck Converter (Sub)
C1
100uF
ESP8266_NodeMCU_V2
DS3231
DS3231
RTC
CLK
Ds
LOAD
Dot
GND



DIAGRAM



DIAGRAM



06

CODE

ส่วนของโค้ดโปรแกรม

CODE

```ino
#include <ESP8266WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <Arduino.h>


// Shift Register SN74HC595N 8-bit SIPO
#define SRCLK D5
#define SER_DATA D6
#define LATCH D7
#define DOTpin D8
uint8_t digits[10] = {126, 48, 109, 121, 51, 91, 95, 112, 127, 123};  // 0 -> 9 from binary
uint8_t celsiusUnit[2] = {0b01100011, 0b01001110};
// 0b 0 X X X X X X X
//       a b c d e f g


// DS3230
#include <Wire.h>
#include <RtcDS3231.h>
RtcDS3231<TwoWire> Rtc(Wire);
// D1 -> SCL, D2 -> SDA
#define SquareWave_pin D0 // D0 -> SQW
uint8_t SquareWave;


const char* ssid = "NatchPai";
const char* password = "powerpay4";

int16_t pullData_hour = 48;
bool connection;
bool onePullQuota = true;
```

CODE

```
int timezone = 7 * 3600;
int dst = 0;
WiFiUDP ntp;
NTPClient timeClient(ntp, "europe.pool.ntp.org", timezone);

unsigned long previousPullDataTimes;
unsigned long previousTimes;
unsigned long previousTimes2;
unsigned long currentTimes;
unsigned long oldLoadTimes;
unsigned long newLoadTimes;

void setup() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  // Serial.begin(9600);

  //Set WIFI
  timeClient.begin();
  WiFi.setAutoReconnect(true);
  WiFi.persistent(true);

  //Set DS3231
  Rtc.Begin();
  Rtc.SetSquareWavePin(DS3231SquareWavePin_ModeClock); //Sets pin mode
  Rtc.SetSquareWavePinClockFrequency(DS3231SquareWaveClock_1Hz); //Sets frequency
  checkStateRTC();
  pinMode(SquareWave_pin, INPUT);

  // Set 74595
  pinMode(LATCH, OUTPUT);
  pinMode(SRCLK, OUTPUT);
  pinMode(SER_DATA, OUTPUT);
  pinMode(DOTpin, OUTPUT);

  TestStart();
}
```

CODE

```
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 }
```

CODE

```
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
```

CODE

```
void autoPullData() {
  if (onePullQuota) {
    updateDate();
    onePullQuota = false;
    previousPullDataTimes = 0;
    display_SET();
  }

  currentTimes = millis();
  if( (currentTimes - previousPullDataTimes) > (pullData_hour * 3600000)) {
    previousPullDataTimes = currentTimes;
    updateDate();
  }

  if (currentTimes < previousTimes) {previousPullDataTimes = 0;}
}

void updateDate() {
  timeClient.update();
  time_t epochTime = timeClient.getEpochTime();
  struct tm *ptm = gmtime ((time_t *) &epochTime);

  uint16_t year = ptm -> tm_year + 1900;
  uint8_t month = ptm -> tm_mon + 1;
  uint8_t dayMonth = ptm -> tm_mday;
  uint8_t hour = timeClient.getHours();
  uint8_t min = timeClient.getMinutes();
  uint8_t sec = timeClient.getSeconds() + 1;  // 1 is Fix delay

  Rtc.SetDateTime(RtcDateTime(year, month, dayMonth, hour, min, sec));
}
```

CODE

```
// Shift Register > below
uint8_t rawSec, rawMinute, rawHour;
uint8_t second_First, second_End;
uint8_t minute_First, minute_End;
uint8_t hour_First, hour_End;

uint8_t partitionFirstDigit(uint8_t data) {
  if( int(data) >= 10) return data / 10;
  else return 0;
}

uint8_t partitionEndDigit(uint8_t data) {
  return int(data) % 10;
}

void analyzeData(RtcDateTime now) {
  rawSec = now.Second();
  rawMinute = now.Minute();
  rawHour = now.Hour();
  // partitionDigit below
  second_First = partitionFirstDigit(rawSec);
  second_End = partitionEndDigit(rawSec);
  minute_First = partitionFirstDigit(rawMinute);
  minute_End = partitionEndDigit(rawMinute);
  hour_First = partitionFirstDigit(rawHour);
  hour_End = partitionEndDigit(rawHour);
}
```

CODE

```
// LatchData(hour_first, hour_last, minute_first, minute_last, second_first, second_last);
void LatchData(uint8_t Q5, uint8_t Q4, uint8_t Q3, uint8_t Q2, uint8_t Q1, uint8_t Q0) {
  digitalWrite(LATCH, 0);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, Q0);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, Q1);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, Q2);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, Q3);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, Q4);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, Q5);
  digitalWrite(LATCH, 1);
  digitalWrite(LATCH, 0);
}

void ResetDisplay() {
  digitalWrite(DOTpin, 0);
  digitalWrite(LATCH, 0);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
  shiftOut(SER_DATA, SRCLK, LSBFIRST, 0b00000000);
  digitalWrite(LATCH, 1);
  digitalWrite(LATCH, 0);
}


// Auto Mode Set Below
uint8_t mode = 1;
uint8_t oldStat;
unsigned int countSquare;
```

CODE

```
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
267
268 void blinkDot() {
269   if (SquareWave == 1) {
270     digitalWrite(DOTpin, 0);
271   }
272   else {
273     digitalWrite(DOTpin, 1);
274   }
275 }
276
```

CODE

```
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
```

จัดทำโดย

นางสาวกฤติมา พัฒน์ซ้าย

นายธนัท ตันอึ๊ง

นายณัฐชนน เถื่อนถ้ำ

ห้อง E31