**Color Matching Game**

**จัดทำโดย**

6330132621 ณฐภัทร แกล้วกล้า

6330271121 นลิน ใบพลูทอง

6330308821 ปรินทร์ โอภาสผาติกุล

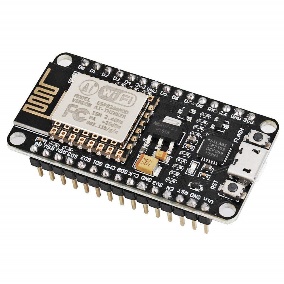
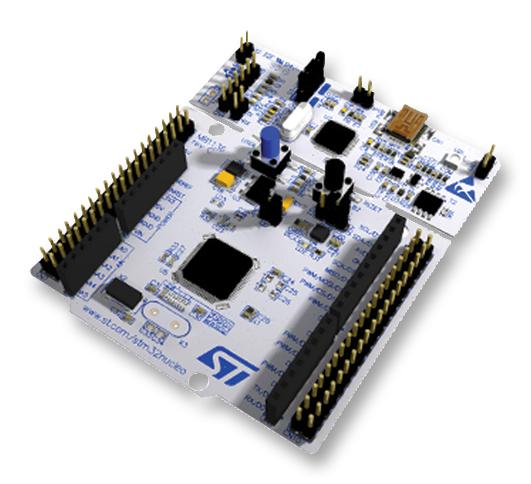
6331308021 ชานน รัตนจรัสกุล

รายงานนี้เป็นส่วนหนึ่งของวิชา

2110366 Embedded System Laboratory (2021/2)

|  |
| --- |
| **นักเชื่อม System Architecture**  6330132621 ณฐภัทร แกล้วกล้า |
| * ออกแบบและพัฒนา database และระบบส่งข้อมูล * เชื่อม STM32 กับ ESP8266 * เชื่อม STM32 กับ wifi * เชื่อม STM32 กับ database ของ Firebase |
| **UI/UX Designer and Development**  6330271121 นลิน ใบพลูทอง |
|  |
| **Website Development**  6330308821 ปรินทร์ โอภาสผาติกุล |
|  |
| **Embedded System Development**  6331308021 ชานน รัตนจรัสกุล |
| * สร้างโค้ดเพื่อให้ Color sensor และ Sound sensor ทำงานได้ * สร้างโค้ดและเชื่อมต่อ Sensor กับ STM32 เพื่อควบคุมการทำงาน และรับส่งข้อมูล * Calibrate color sensor เพื่อให้อ่านค่าสีได้ตรงกับความจริงมากที่สุด * เขียน report ในหัวข้อที่เกี่ยวข้อง |

**บทบาทหน้าที่ของสมาชิก**

******อุปกรณ์ที่ใช้**

**Sound sensor**

**NodeMCU**

**STM32**

**Website**

**UX/UI**

* STM32 NUCLEO-F411RE

**Color sensor**

* ESP8266 (NodeMCU)
* Color Sensor
* Sound Sensor
* Jumper wires female to female (12 เส้น)

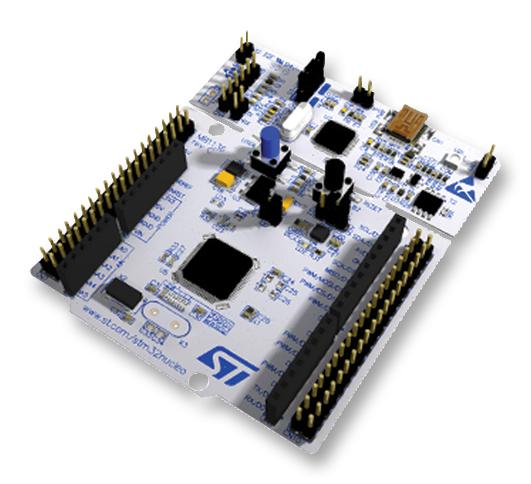
****

A picture containing text, electronics

Description automatically generated**การเชื่อมต่อ**

**วิธีใช้งาน**

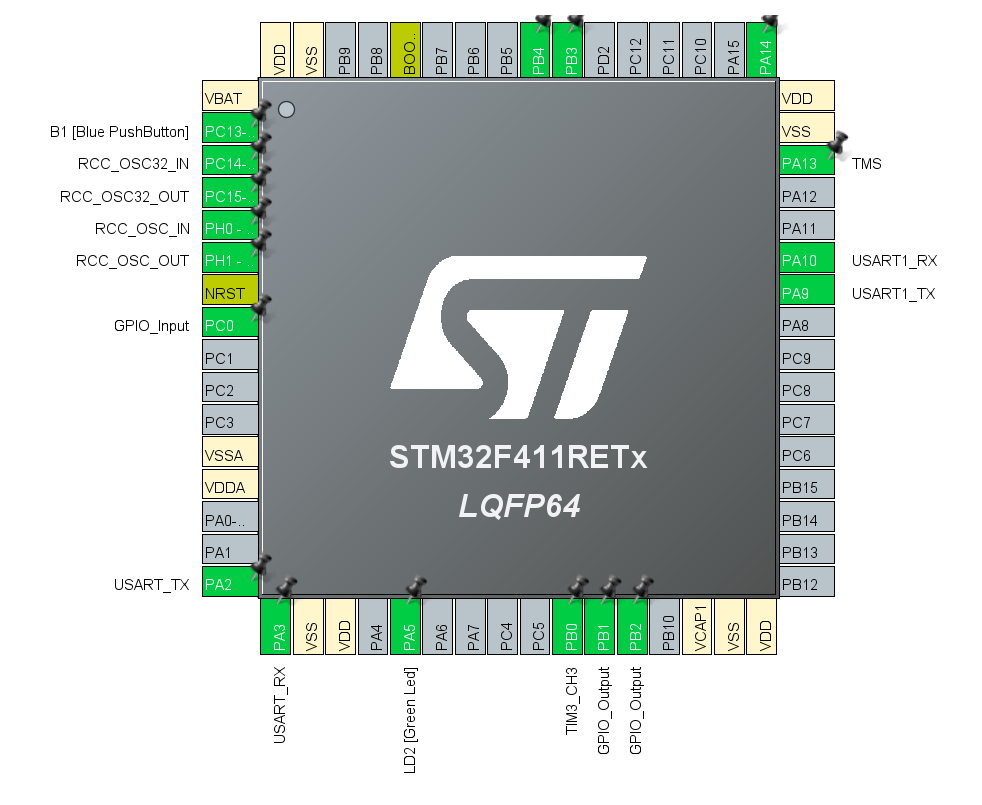
เมื่อเริ่มเกมจะมีเวลาทั้งหมด 2 นาที เว็บไซต์จะสุ่มสีขึ้นมาสีหนึ่ง จากนั้นผู้เล่นจะต้องหาสิ่งของใดๆ ก็ได้ที่มีสีเหมือน หรือใกล้เคียงสีที่เว็บไซต์สุ่มขึ้นมา เมื่อหาสิ่งของได้แล้ว ให้ผู้เล่นนำ color sensor มาวางบนผิวสิ่งของนั้น แล้วตบมือ 3 ครั้ง เพื่อเป็นการบอกให้เว็บไซต์ทำการตรวจสอบสี หากสีที่ได้ใกล้เคียงสีที่เว็บไซต์สุ่มขึ้นมาก็จะได้คะแนน จากนั้นเว็บไซต์ก็จะสุ่มสีใหม่ขึ้นมา แต่หากสีที่ได้ไม่ใกล้เคียงก็จะต้องหาสิ่งของใหม่เพื่อนำมาตรวจสอบ โดยผู้เล่นต้องพยายามหาสิ่งของที่มีสีใกล้เคียงกับสีที่เว็บไซต์สุ่มขึ้นมาให้ได้หลายครั้งมากที่สุด เพื่อให้ได้คะแนนมากที่สุด

**STM32 (NUCLEO-F411RE)**

ใช้ในการเชื่อมต่อกับ Sound sensor, Color sensor และ NodeMCU เพื่อควบคุมการทำงาน และรับส่งข้อมูลระหว่างกัน

**คำอธิบายตัวแปรและฟังก์ชั่นต่างๆ ในไฟล์ main.c**

|  |  |
| --- | --- |
| **ชื่อ** | **คำอธิบาย** |
| **TIMCLOCK** | ค่า HCLK ใน Clock Configuration |
| **PRESCALAR** | ค่า Prescaler ใน TIM 3 (ใช้ในการคำนวณความถี่จาก Color sensor) |
| **RED\_a**  **RED\_b**  **GREEN\_a**  **GREEN\_b**  **BLUE\_a**  **BLUE\_b** | ค่าในการคำนวณเพื่อแปลงความถี่จาก color sensor ไปเป็น ค่าสี RGB  ปรับเปลี่ยนค่าเหล่านี้เพื่อ calibrate sensor |
| **enum Scaling** | Output frequency scaling ของ color sensor (0%, 2%, 20%, 100%) |
| **enum Filter** | Photodiode type ของ color sensor ใช้ในการเลือกสีที่ต้องการวัดค่า RGB (Red, Blue, Clear, Green) |
| **uint8\_t set\_color** | สีที่ต้องการวัดค่า RGB (Red, Blue, Clear, Green) |
| **int wait\_for\_callback** | ใช้บอกว่า TIM 3 วัดความถี่จาก color sensor เสร็จเรียบร้อยแล้วหรือไม่ โดยเมื่อ **HAL\_TIM\_IC\_CaptureCallback** ถูกเรียกเป็นครั้งที่ 2 แสดงว่าวัดความถี่เสร็จเรียบร้อย |
| **double frequency** | ความถี่จาก color sensor |
| **double Output\_Color** | ค่า RGB ของสี **set\_color** (0 – 255) ที่ได้จากการแปลงความถี่จจาก color sensor |
| **double RGB[3]** | เก็บ **Output\_Color** ของสี Red Green Blue ตามลำดับ |
| **uint32\_t IC\_Val1** | ค่าเวลาที่ rising edge ของคลื่นความถี่จาก color sensor ถูกตรวจจับ ครั้งที่ 1 |
| **uint32\_t IC\_Val2** | ค่าเวลาที่ rising edge ของคลื่นความถี่จาก color sensor ถูกตรวจจับ ครั้งที่ 2 |
| **uint32\_t Difference** | ผลต่าง **IC\_Val1** และ **IC\_Val2** |
| **int Is\_First\_Captured** | ใช้จำว่าเป็น rising edge ครั้งที่ 1 ได้ถูกตรวจจับแล้วหรือไม่ |
| **void HAL\_TIM\_IC\_CaptureCallback(TIM\_HandleTypeDef \*htim)** | ใช้วัดความถี่จาก color sensor และแปลงเป็นค่า **Output\_Color** |
| **void Set\_Scaling(int mode)** | ปรับเปลี่ยนค่า Pin S0, S1 ของ color sensor ตาม output frequency scaling ที่ต้องการ |
| **void Set\_Filter(uint8\_t mode)** | ปรับเปลี่ยนค่า Pin S2, S3 ของ color sensor ตามสี Red/Green/Blue ที่ต้องการอ่าน |
| **void Print\_Output()** | ใช้ส่งข้อมูล **RGB[3]** ผ่าน UART1 ไปยัง NodeMCU |
| **void Print\_Frequency(uint8\_t set\_color, float sum\_frequency)** | ใช้แสดงข้อมูล ค่า RGB และ ค่า frequency ทั้ง 3 สี ผ่าน UART2 เพื่อใช้ในการ calibrate sensor |
| **float GetColor(uint8\_t set\_color)** | **Set\_Filter** เป็นสีที่ต้องการอ่าน จากนั้นเปิดการ interrupt ของ TIM3 เพื่อให้ **HAL\_TIM\_IC\_CaptureCallback** ถูกเรียก จนวัดความถี่จาก color sensor เสร็จ แล้วจึงปิดการ interrupt และคืนค่า **Output\_Color** |
| **void ReadColor(int read\_times)** | ให้ color sensor อ่านค่าสี Red Green Blue ทั้งหมด **read\_times** ครั้ง แล้วคิดค่าเฉลี่ย แล้วบันทึกลงใน **RGB[3]** |
| **void ReadColorWithFrequency(int read\_times, int delay)** | ให้ color sensor อ่านค่าสี Red Green Blue รวมทั้ง frequency ทั้งหมด **read\_times** ครั้ง แล้วคิดค่าเฉลี่ย จากนั้นแสดงผลผ่าน **Print\_Frequency** ไปเรื่อยๆ ทุกๆ **delay** มิลลิวินาที เพื่อใช้ในการ calibrate sensor |
| **int main(void)** | ใช้อ่านค่าจาก sound sensor ว่าตรวจจับเสียงหรือไม่ ถ้าตรวจจับเสียงต่อเนื่องกัน 3 ครั้ง โดยแต่ละครั้งห่างกันไม่เกิน 1 วินาที จะสั่งให้ **ReadColor** และ **Print\_Output**   * มีการ debounce เพื่อป้องกันการตรวจจับเสียงหลายครั้ง จากการส่งเสียง 1 ครั้ง โดยไม่นับเสียงหากได้รับเสียงต่อกันเร็วกว่า 20 ms * ป้องกันการนับเสียงหลายครั้ง หากส่งเสียงติดต่อกัน โดยตรวจสอบว่าก่อนการตรวจจับเสียง จะต้องมีช่วงที่ไม่ตรวจจับเสียงก่อน |

**Pinout & Configuration**

**Diagram, schematic

Description automatically generatedClock Configuration**

**Sound sensor**

ใช้ในการตรวจจับเสียง เมื่อผู้เล่นต้องการเกมทำการตรวจสี ให้ปรบมือ 3 ครั้ง โดยระยะห่างแต่ละครั้งไม่เกิน 1 วินาที สาเหตุที่ให้ปรบมือถึง 3 ครั้ง เพราะเพื่อป้องกันผู้เล่นส่งเสียงโดยไม่ได้ตั้งใจ จากนั้นจะสั่งให้ Color sensor วัดสีต่อไป

* มี LED สีเขียวที่ sensor 2 ตัว ตัวแรกจะติดเมื่อเชื่อมต่อ sensor กับ STM32 ตัวที่ 2 จะติดเมื่อได้รับเสียงที่ดังมากกว่าที่ตั้งไว้
* สามารถตั้งระดับเสียงที่ทำให้ Sound sensor ตรวจจับได้ โดยการหมุนสวิตซ์ที่มีรูปร่าง +
* เมื่อ sensor ตรวจจับเสียง GPIOC PIN 1 จะมีค่า HAL\_OK

**วิธีการต่อเข้ากับ STM32**

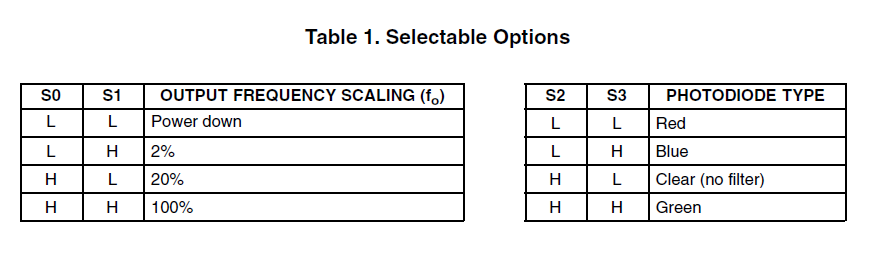
|  |  |  |
| --- | --- | --- |
| **Sound sensor** | **STM32** | **Pinout configuration** |
| OUT | PC1 | GPIO\_Input |
| VCC | 3V |  |
| GND | GND |  |

**แหล่งที่มา**

* https://www.micropeta.com/video41
* https://www.youtube.com/watch?v=CN0sRkJhPXE

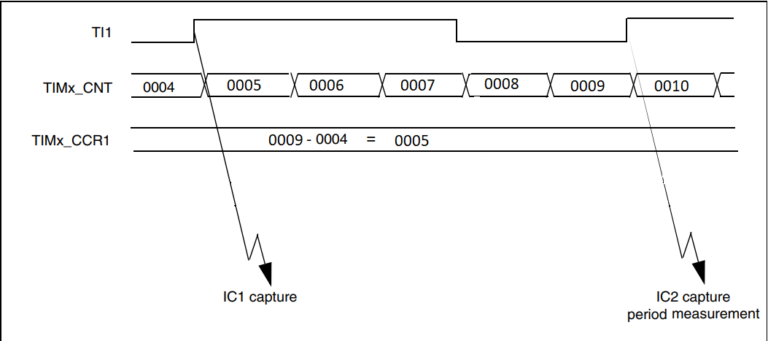
**Color sensor (TCS3200)**

ใช้ในการวัดสีของสิ่งของต่างๆ โดยจะได้ผลลัพธ์ออกมาเป็นความถี่ จากนั้นต้องแปลงความถี่ที่ได้เป็นค่า RGB 0-255 โดยการวัดสีแต่ละครั้งจะสามารถวัดได้ทีละ 1 สี (Red/Green/Blue)

* มี LED สีขาวที่ sensor 4 ตัว จะติดเมื่อเชื่อมต่อ sensor กับ STM32 มีหน้าที่เพิ่มความสว่างให้กับผิวของสิ่งของที่ต้องการวัด
* สามารถปรับ Scale ของความถี่ที่ได้ โดยตั้งค่า Pin S0, S1 ดังตาราง
* Sensor สามารถอ่านค่าได้ทีละสี (Red/Green/Blue) โดยตั้งค่า Pin S2, S3 เพื่อเลือก Filter สีที่จะอ่าน
* วัดความถี่โดยใช้ TIM3\_CH3 โดยตั้ง

Clock Source = Internal Clock

Channel3 = Input Capture direct mode

NVIC Settings: TIM3 global interrupt = Enabled

* วัดความถี่โดยนับระยะเวลาระหว่าง Rising edge 2 ครั้ง นั่นคือระยะเวลาระหว่าง การเรียกฟังก์ชั่น HAL\_TIM\_IC\_CaptureCallback 2 ครั้ง และหาผลต่าง
* ใช้ logarithmic equation ในการแปลงความถี่เป็นสี RGB คือ

ค่า RGB ของสีแดง เขียว น้ำเงินตามลำดับ มีค่าตั้งแต่ 0 – 255 (หากคำนวณได้น้อยกว่า 0 หรือมากกว่า 255 ให้ เปลี่ยนเป็น 0 หรือ 255)

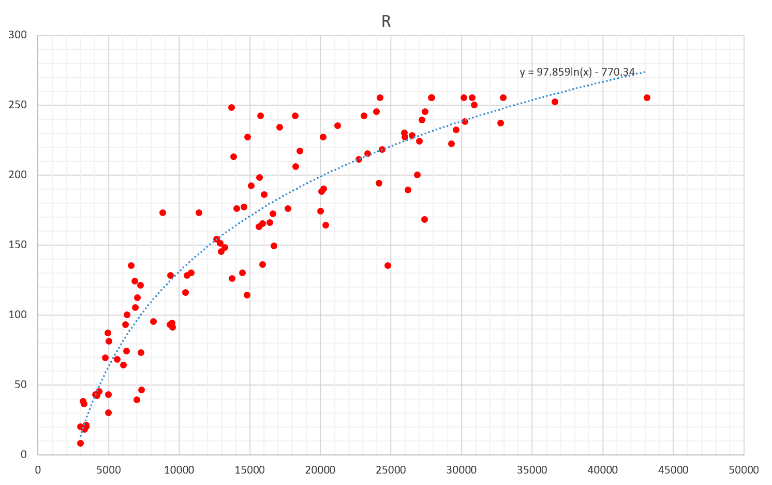
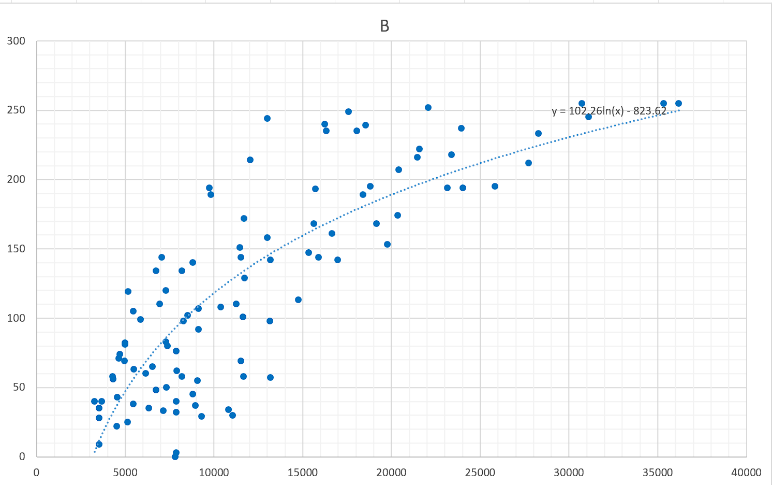
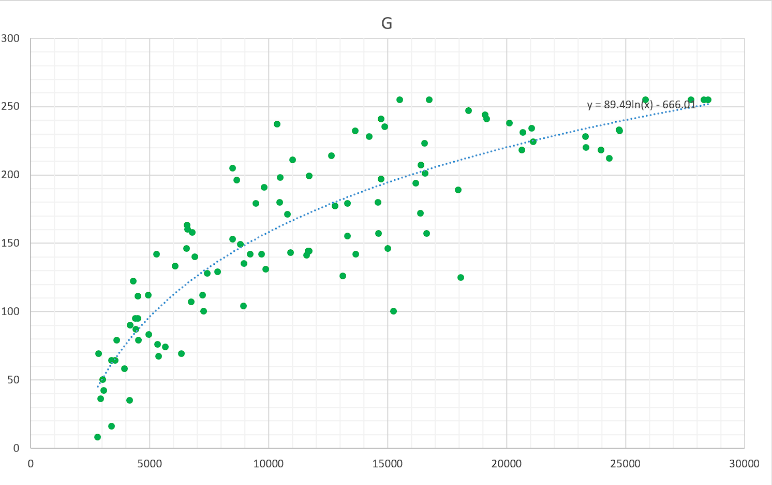
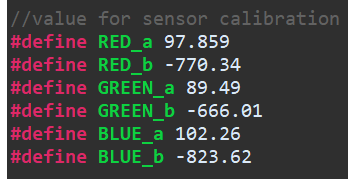
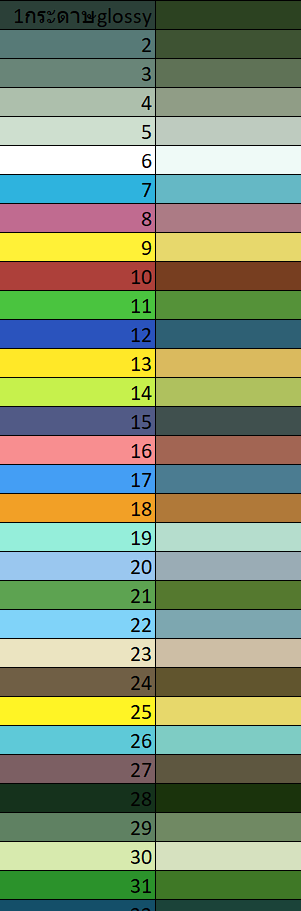
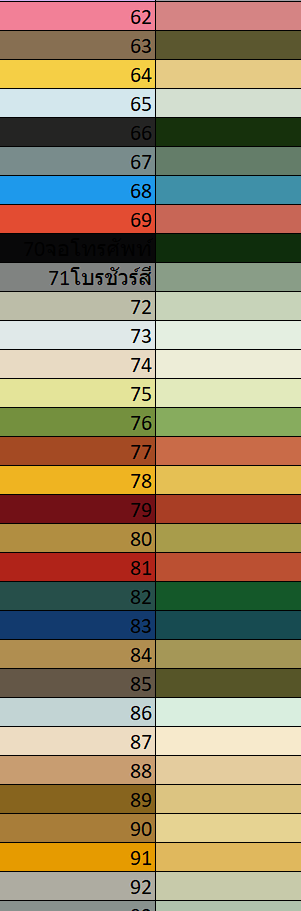
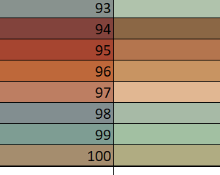
*ของสีแดง เขียว น้ำเงินตามลำดับ*

**Sensor calibration**

1. เก็บข้อมูลความถี่ที่วัดได้จาก sensor และ ค่า RGB ของสิ่งของที่วัด

* ใช้แอพ ColorPicker เพื่อหาค่า RGB ของสิ่งของ
* ใช้เว็บไซต์ https://www.w3schools.com/colors/colors\_rgb.asp เพื่อตรวจสอบและปรับสีได้ได้จากแอพให้ตรงกับความจริง
* A picture containing chart

  Description automatically generatedเก็บข้อมูลทั้งหมด 100 ครั้ง สามารถอ่านเพิ่มได้ในไฟล์ sensor\_calibrate.xlsm

1. นำความถี่ และ ค่า RGB มาสร้างกราฟ และหาสมการในการแปลงความถี่เป็น RGB ด้วย คำสั่ง Trendline ของโปรแกรม Excel
2. พบว่า logarithmic equation มีแนวโน้มของกราฟใกล้เคียงกับข้อมูลมากที่สุด จึงเลือกใช้
3. สามารถปรับค่าสัมประสิทธิ์ของ logarithmic equation ที่ #define ใน main.c
4. ได้ผลลัพธ์ที่ใกล้เคียงสีของสิ่งของ จากภาพสีในช่องด้านซ้ายคือสีของสิ่งของ ส่วนด้านขวาคือสีที่ได้จากการคำนวณจากสมการแปลงความถี่เป็นสี RGB

**วิธีในการตรวจสอบว่าสีคล้ายกันหรือไม่**

* https://stackoverflow.com/questions/25168445/how-to-determine-if-a-color-is-close-to-another-color
* https://stackoverflow.com/questions/27374550/how-to-compare-color-object-and-get-closest-color-in-an-color/27375621#27375621

พบว่าควรตรวจสอบโดย

ถ้า

1. (ผลต่างค่า R)2 +(ผลต่างค่า G)2 + (ผลต่างค่า B)2 <= 5000 หรือ
2. ผลต่างค่า Hue <= 20 หรือ >= 340

แสดงว่าสีคล้ายกัน

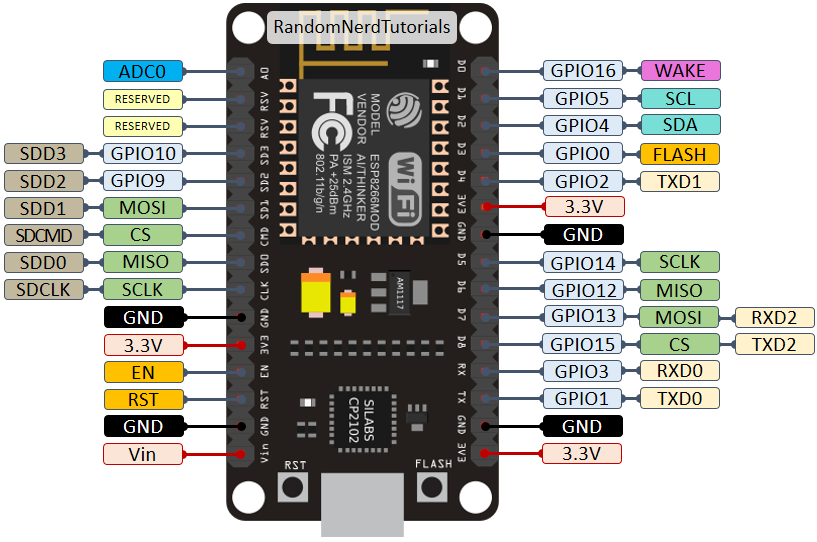
**วิธีการต่อเข้ากับ STM32**

|  |  |  |
| --- | --- | --- |
| **Color sensor** | **STM32** | **Pinout configuration** |
| OUT | PB0 | TIM3\_CH3 |
| S0 | PB1 | GPIO\_Output |
| S1 | PB2 | GPIO\_Output |
| S2 | PB3 | GPIO\_Output |
| S3 | PB4 | GPIO\_Output |
| VCC | 5V |  |
| GND | GND |  |

**แหล่งที่มา**

* https://github.com/jaimelaborda/TCS3200\_STM32F4\_Library
* https://www.mouser.com/catalog/specsheets/tcs3200-e11.pdf
* https://controllerstech.com/input-capture-in-stm32/
* https://deepbluembedded.com/stm32-gpio-tutorial/

**NodeMCU (ESP8266)**



ใช้การรับส่งข้อมูลระหว่าง STM32, NodeMCU และ Firebase เมื่อรับข้อมูลมาจนถึง ‘\0’ จะทำการส่งข้อมูลต่อไปให้ Firebase ผ่าน Wi-Fi ที่กำหนดชื่อและรหัสไว้แล้ว

**วิธีการต่อเข้ากับ STM32**

|  |  |  |
| --- | --- | --- |
| **NodeMCU** | **STM32** | **Pinout configuration** |
| TX(GPIO1) | PA10 | UART1\_RX |
| RX(GPIO3) | PA9 | UART1\_TX |

**แหล่งที่มา**

* https://diyi0t.com/uart-tutorial-for-arduino-and-esp8266/

**Firebase**

ใช้การจัดเก็บใน database ของ firebase ตามรูป

เมื่อข้อมูลถูกส่งมาจาก NodeMCU ข้อมูลจะถูกจัดเก็บใน database และ website จะดึงข้อมูลจาก firebase ไปแสดงผล

**แหล่งที่มา**

* https://randomnerdtutorials.com/esp8266-data-logging-firebase-realtime-database/

**UX/UI**

**Website**

**Picture Credit**

https://www.eradel.com/wp-content/uploads/2019/04/AD307-21.jpg

https://inwfile.com/s-fa/u4skz1.jpg

https://th.element14.com/productimages/large/en\_GB/2433469-40.jpg

https://m.media-amazon.com/images/I/617T2JKnxiL.\_SL1000\_.jpg

https://www.codebee.co.th/labs/wp-content/uploads/2017/03/firebase-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3.png

<https://cdn-icons.flaticon.com/png/512/3518/premium/3518229.png?token=exp=1653039138~hmac=e829eefb975b1b8c84ca7bdfb8a3b850>

<https://cdn-icons-png.flaticon.com/512/922/922699.png>

<https://github.com/jaimelaborda/TCS3200_STM32F4_Library/raw/master/wiki/modes_and_filter.PNG>

https://controllerstech.com/wp-content/uploads/2021/09/IC\_7-768x341.png