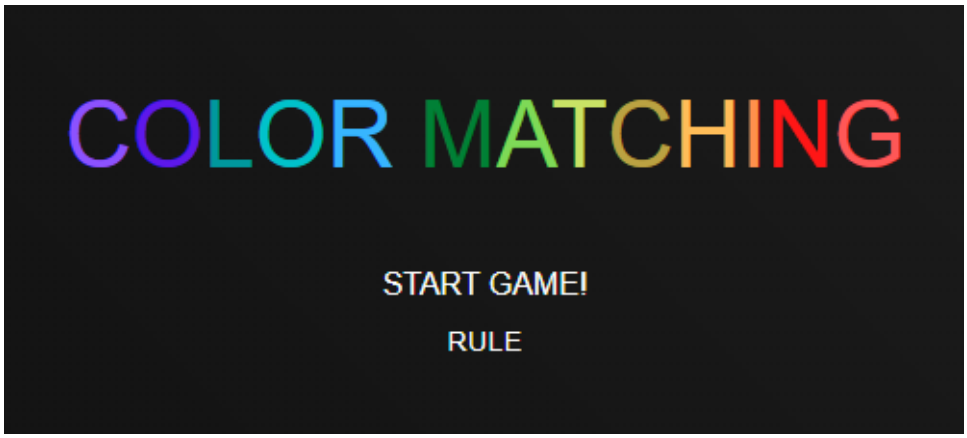
Color Matching Game



จัดทำโดย

6330132621 ณฐภัทร แกล้วกล้า

6330271121 นลิน ใบพลูทอง

6330308821 ปรินทร์ โอภาสผาติกุล

6331308021 ชานน รัตนจรัสกุล

รายงานนี้เป็นส่วนหนึ่งของวิชา

2110366 Embedded System Laboratory (2021/2)

บทบาทหน้าที่ของสมาชิก

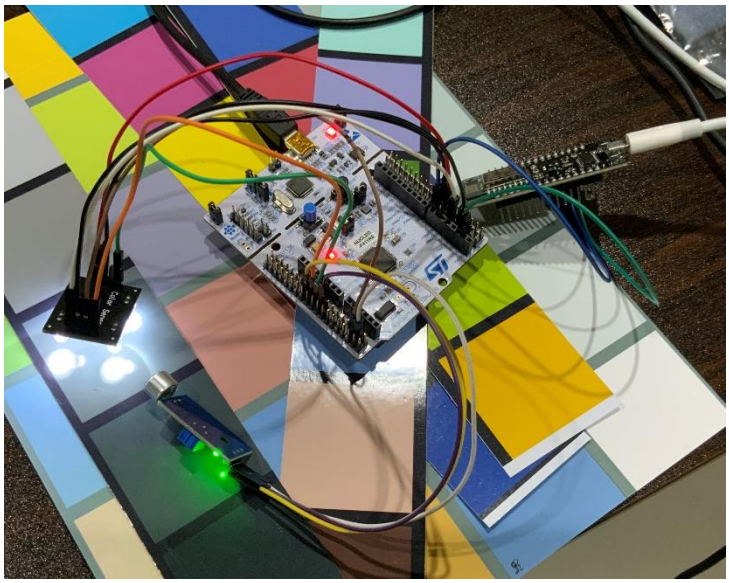
System Architecture 6330132621 ณฐภัทร แกล้วกล้า
- ออกแบบและพัฒนา database และระบบส่งข้อมูล - เชื่อม STM32 กับ ESP8266 - เชื่อม ESP8266 กับ wifi - เชื่อม STM32 และ ESP8266 กับ Firebase
UI/UX Designer and Development 6330271121 นลิน ใบพลูทอง
- ออกแบบหน้าตา website - พัฒนา website - ออกแบบ logic การเล่นเกมใน website
Website Development 6330308821 ปรินทร์ โอภาสผาติกุล
- พัฒนา websie - เชื่อม Firebase กับ website - Deploy website - ออกแบบ logic การเล่นเกมใน website
Embedded System Development 6331308021 ชานน รัตนจรัสกุล
- สร้างโค้ดเพื่อให้ Color sensor และ Sound sensor ทำงานได้ - สร้างโค้ดและเชื่อมต่อ Sensor กับ STM32 เพื่อควบคุมการทำงาน และรับส่งข้อมูล - Calibrate color sensor เพื่อให้อ่านค่าสีได้ตรงกับความจริงมากที่สุด

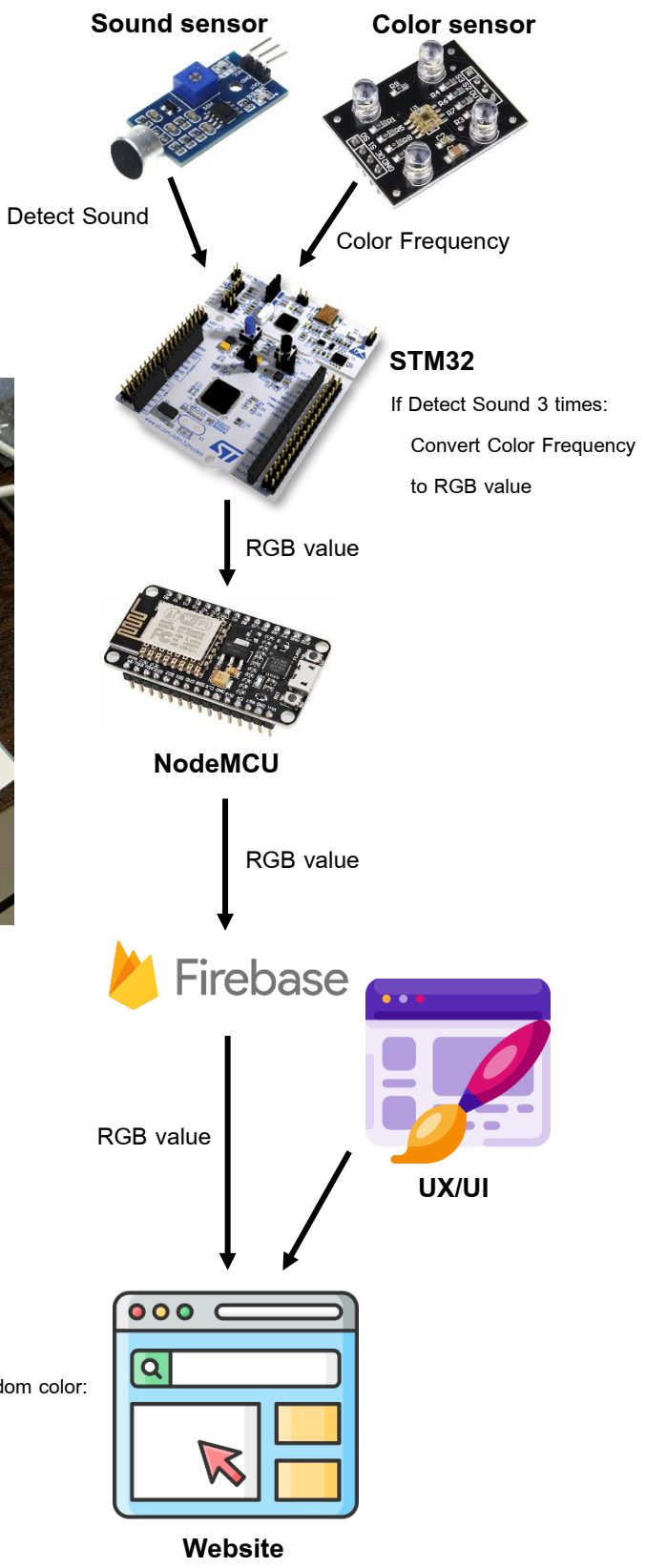
GitHub: https://github.com/NonRoute/2110366_EMBEDDED_SYS_LAB_I_Project

Website: https://musical-nasturtium-4674ad.netlify.app/

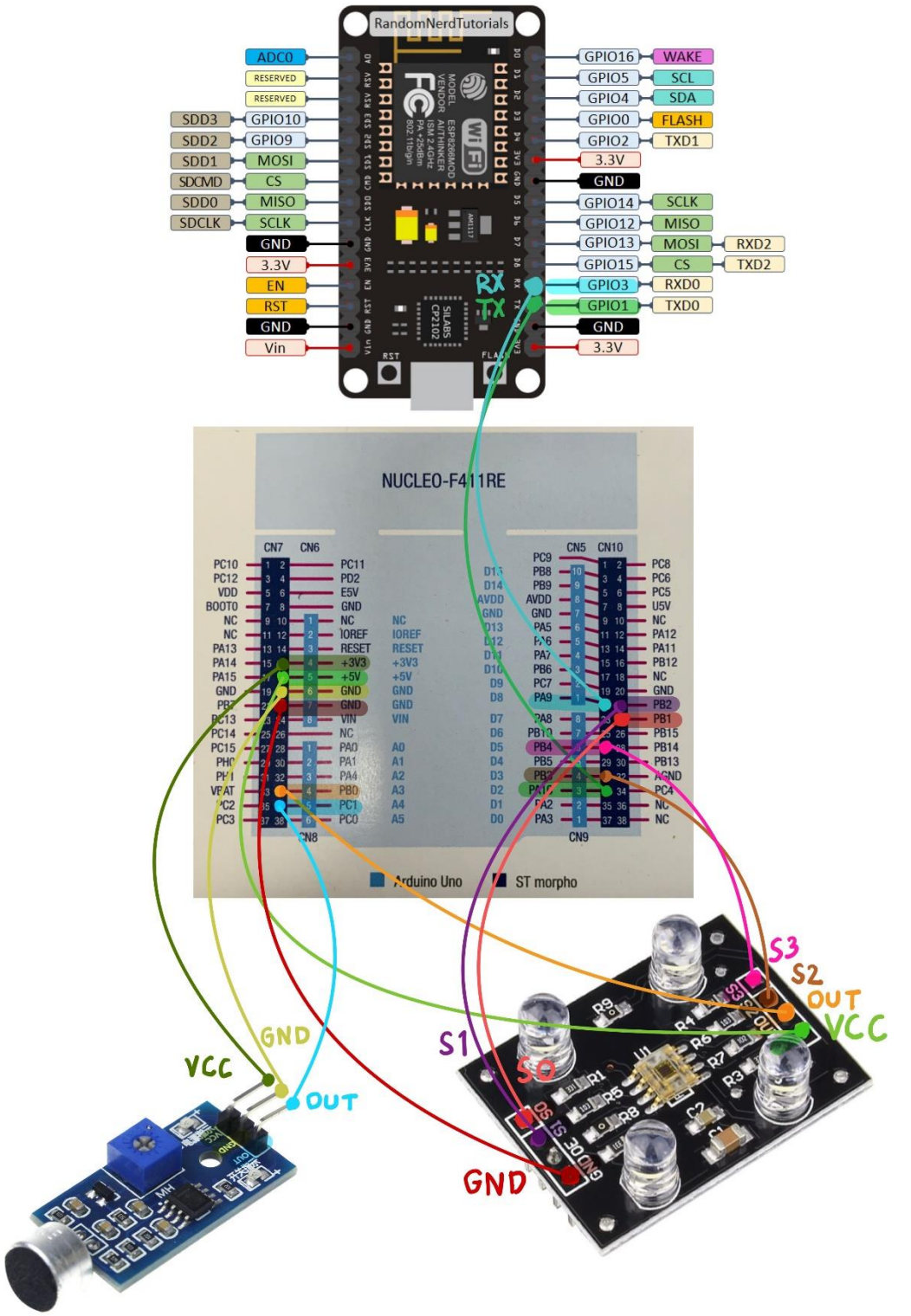
อุปกรณ์ที่ใช้

- STM32 NUCLEO-F411RE
- ESP8266 (NodeMCU)
- Color Sensor
- Sound Sensor
- Jumper wires female to female (12 เส้น)





การเชื่อมต่อ


RandomNerdTutorials
ADC0
RESERVED
RESERVED
SDD3
GPIO10
SDD2
GPIO9
SDD1
MOSI
SDCMD
CS
SDD0
MISO
SDCLK
SCLK
GND
3.3V
EN
RST
GND
Vin
GPIO16
WAKE
GPIO5
SCL
GPIO4
SDA
GPIO0
FLASH
GPIO2
TXD1
3.3V
GND
GPIO14
SCLK
GPIO12
MISO
GPIO13
MOSI
RXD2
GPIO15
CS
TXD2
GPIO3
RXD0
GPIO1
TXD0
GND
3.3V
RX
TX
NUCLEO-F411RE
CN7
CN6
CN5
CN10
CN8
CN9
Arduino Uno
ST morpho
VCC
GND
OUT
S1
S2
S3
OUT
VCC
GND

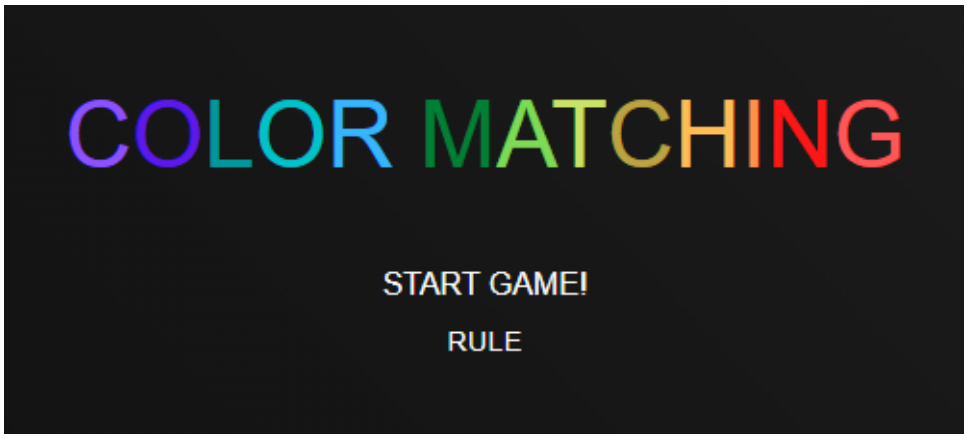
วิธีใช้งาน

กฎการเล่นเกม

เมื่อเริ่มเกมจะมีเวลานับถอยหลังทั้งหมด 2 นาที เว็บไซต์จะสุ่มสีขึ้นมาหนึ่งสี ผู้เล่นจะต้องหาสิ่งของใดๆ ที่มีสีเหมือนกับสีที่เว็บไซต์สุ่มขึ้นมา เมื่อหาสิ่งของได้แล้วให้ผู้เล่นนำมาวางหน้า color sensor แล้วปรบมือ 3 ครั้ง เพื่อเป็นการบอกให้เว็บไซต์ทำการตรวจสอบสี หากสีที่ได้ใกล้เคียงสีที่เว็บไซต์สุ่มขึ้นมาก็จะได้คะแนน จากนั้นเว็บไซต์ก็จะสุ่มสีใหม่ขึ้นมาแล้วเก็บคะแนนต่อไปเรื่อยๆ แต่หากสีของสิ่งของที่หามานั้นไม่ใกล้เคียง ก็จะต้องหาของใหม่เพื่อนำมาตรวจสอบ โดยผู้เล่นต้องพยายามหาสิ่งของที่มีสีใกล้เคียงกับสีที่เว็บไซต์สุ่มขึ้นมาให้ได้หลายครั้งมากที่สุด เพื่อเก็บคะแนนให้ได้เยอะที่สุดภายในเวลาที่กำหนด โดยมีปุ่มเพื่อให้ข้ามสีที่ไม่ต้องการได้ แต่จะเสียคะแนนแทน

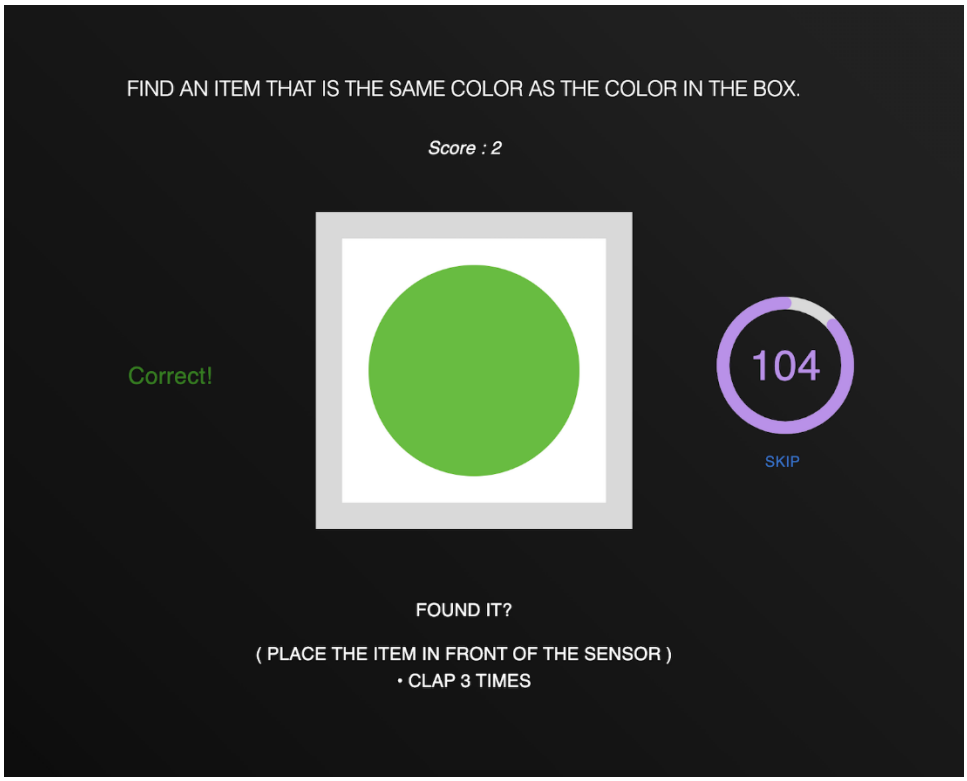
วิธีการเล่นเกม

เริ่มเข้าหน้าเกมจะมีให้กด 2 ส่วนคือปุ่ม start game กับ rule หากผู้เล่นยังไม่เข้าใจกฎการเล่นเกม สามารถกดปุ่ม rule เพื่อเข้าไปอ่านวิธีการเล่นเกมก่อนได้ เมื่อพร้อมแล้ว ให้กดปุ่ม start game เพื่อเริ่มเกม

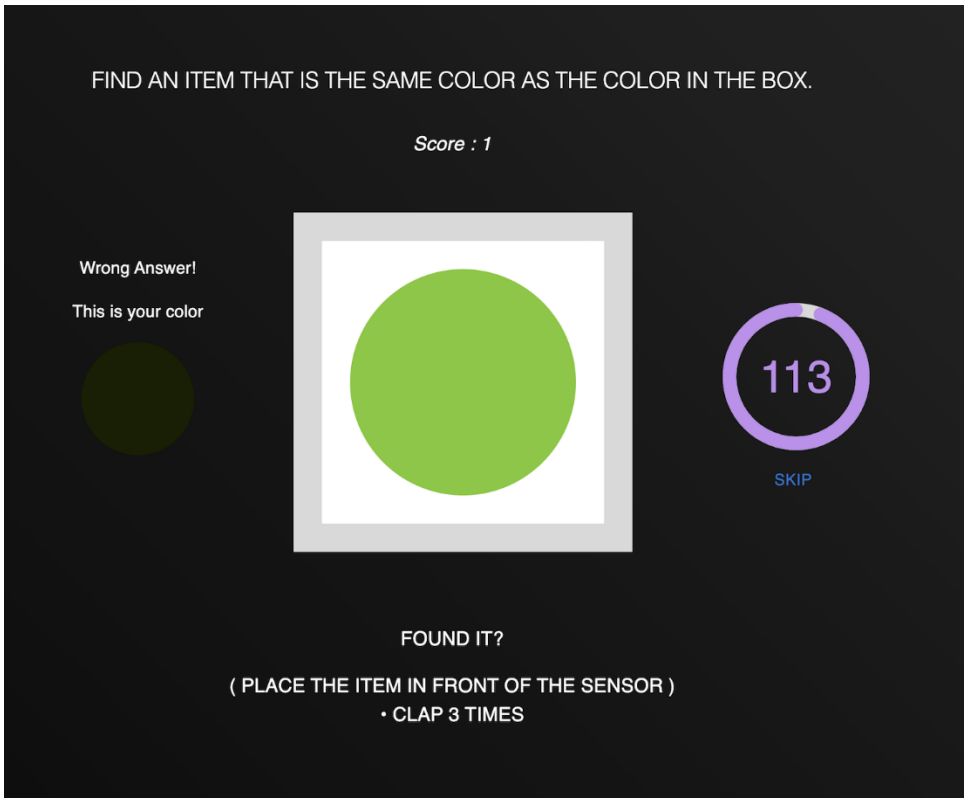


ในเกมจะมีเวลานับถอยหลังให้เห็นทางฝั่งขวา เป็นเวลา 120 วินาที
ผู้เล่นต้องหาสิ่งของที่มีสีคล้ายกับสีที่ปรากฏอยู่บนหน้าเว็บไซต์ แล้วจึงนำสิ่งของที่หาได้นั้นมาวางไว้ที่หน้า color sensor

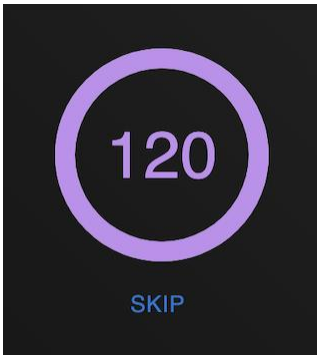
ถ้าสีของสิ่งของนั้นเหมือนกับสีที่เว็บไซต์กำหนด จะขึ้นข้อความ correct ทางฝั่งซ้าย และ score จะถูกเพิ่มขึ้นอีก 1 คะแนน



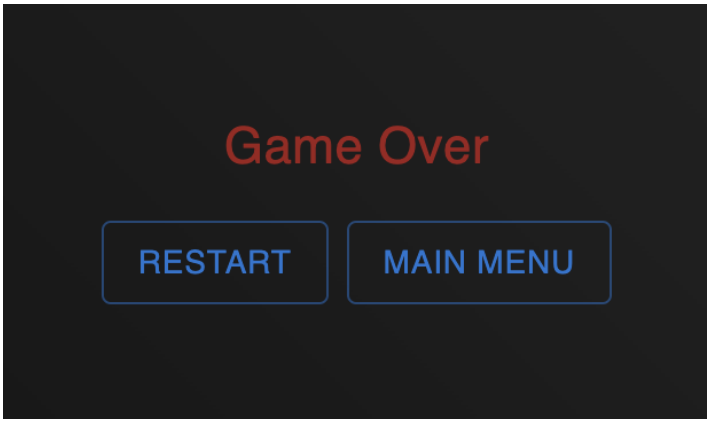
แต่ถ้าสีนั้นไม่ใกล้เคียงกับสีที่เว็บไซต์กำหนด จะขึ้นเตือนว่าสีที่หามานั้นผิด



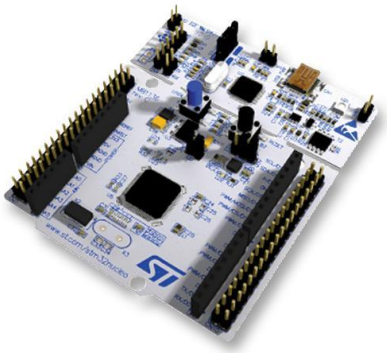
ถ้าผู้เล่นต้องการข้ามสี สามารถกดปุ่ม skip ตรงฝั่งขวาได้ แต่ score จะถูกลดลง 1 คะแนนทุกครั้งที่กดปุ่ม skip



เมื่อหมดเวลาจะสามารถกดปุ่ม restart เพื่อเล่นเกมอีกรอบ หรือกดปุ่ม main manu เพื่อกลับไปยังหน้าหลักได้



STM32 (NUCLEO-F411RE)



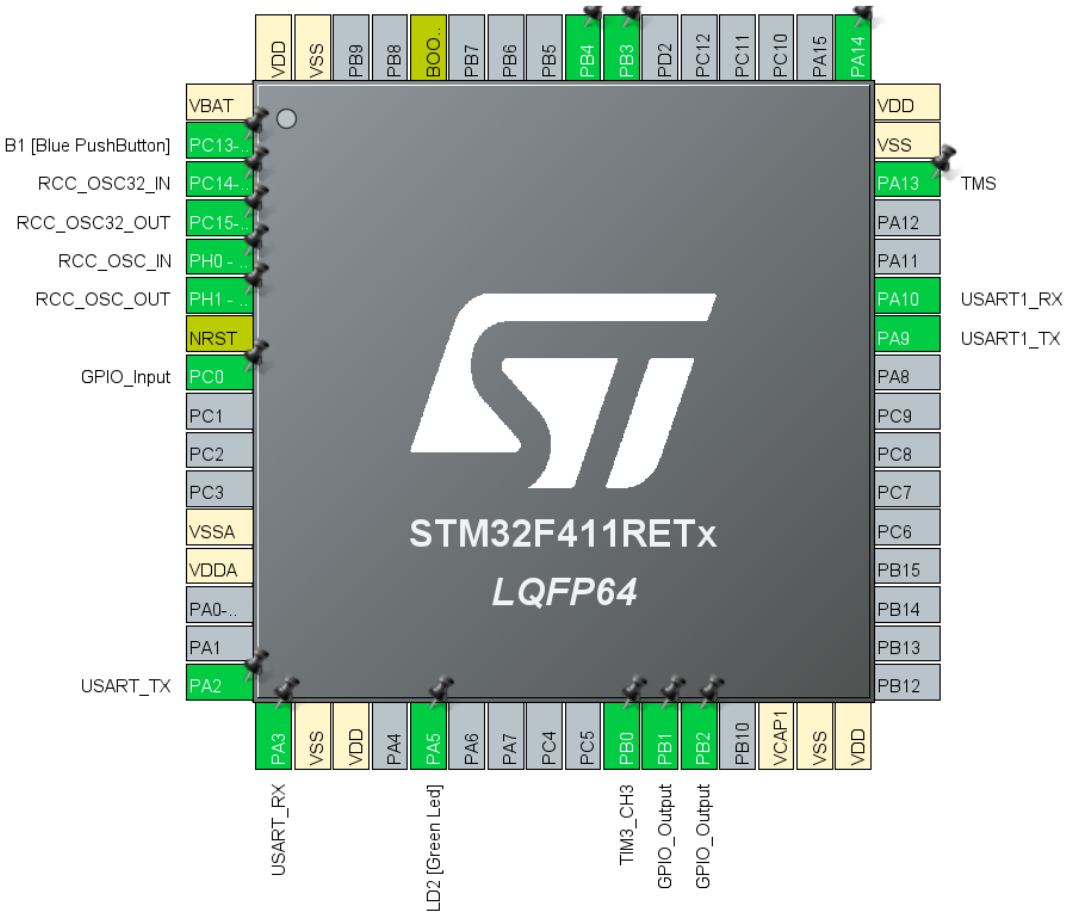
ใช้ในการเชื่อมต่อกับ sound sensor, color sensor และ nodeMCU เพื่อควบคุมการทำงาน และรับส่งข้อมูลระหว่างกัน โดยเมื่อ sound sensor ตรวจจับเสียงต่อเนื่องกัน 3 ครั้ง จะสั่งให้ color sensor วัดสีของสิ่งของ จากนั้นส่งค่าสี RGB ไปยัง nodeMCU ต่อไป โดยจะ toggle LED บน STM32 เพื่อแสดงให้ทราบว่า color sensor ได้ถูกสั่งให้วัดสีแล้ว

คำอธิบายตัวแปรและฟังก์ชั่นต่าง ๆ ในไฟล์ main.c

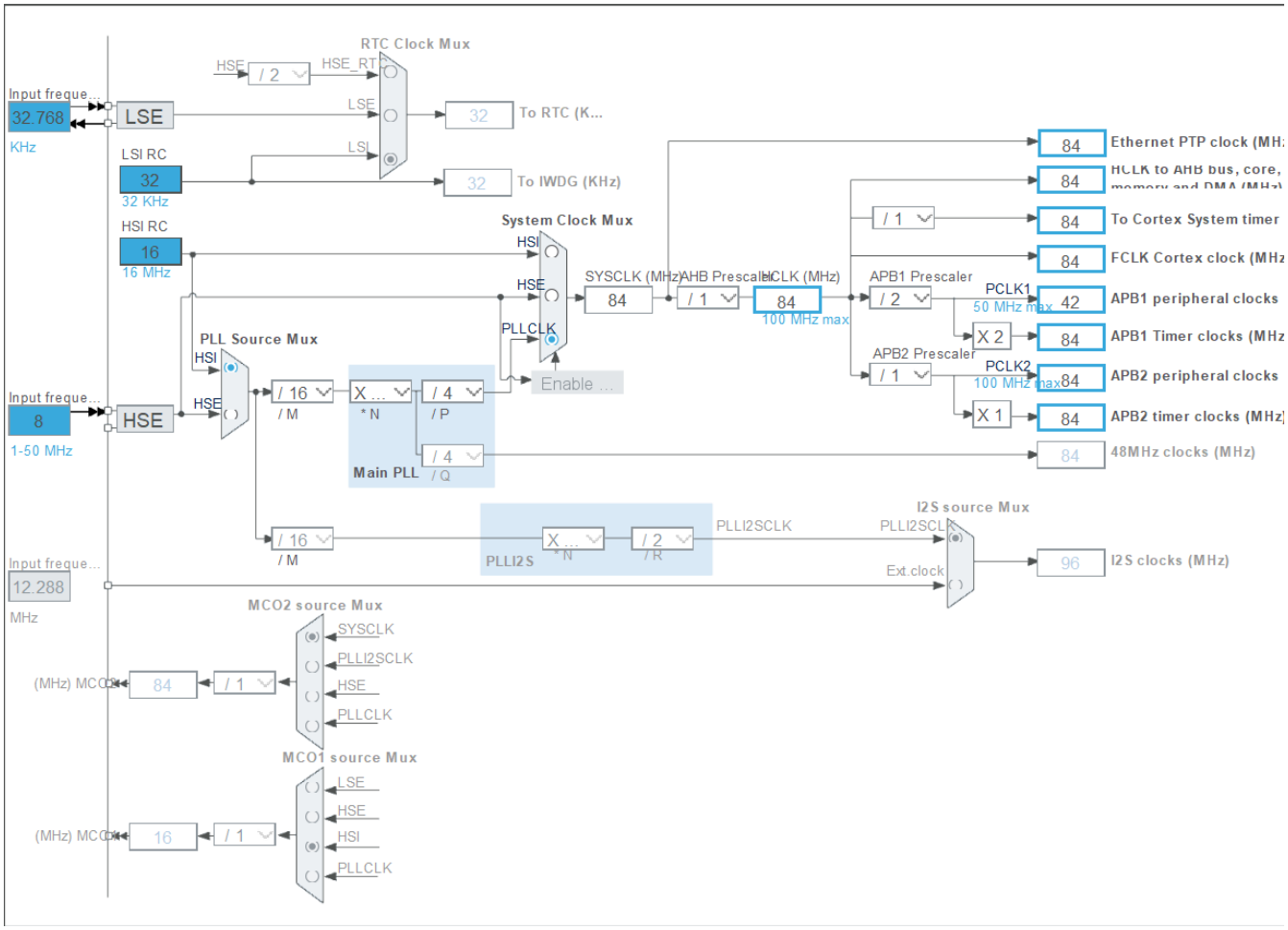
ชื่อ	คำอธิบาย
TIMCLOCK	ค่า HCLK ใน Clock Configuration
PRESCALER	ค่า Prescaler ใน TIM 3 (ใช้ในการคำนวณความถี่จาก color sensor)
RED_a **RED_b** **GREEN_a** **GREEN_b** **BLUE_a** **BLUE_b**	ค่าในการคำนวณเพื่อแปลงความถี่จาก color sensor ไปเป็น ค่าสี RGB ปรับเปลี่ยนค่าเหล่านี้เพื่อ calibrate sensor
enum 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) ที่ได้จากการแปลงความถี่จาก 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 เสร็จแล้วจึงปิดการ 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, Print_Output** และ Toggle LED บน STM32 - มีการ debounce เพื่อป้องกันการตรวจจับเสียงหลายครั้ง จากการส่งเสียง 1 ครั้ง โดยไม่นับเสียงหากได้รับเสียงต่อกันเร็วกว่า 20 ms - ป้องกันการนับเสียงหลายครั้ง หากส่งเสียงติดต่อกัน โดยตรวจสอบว่าก่อนการตรวจจับเสียง จะต้องมีช่วงที่ไม่ตรวจจับเสียงก่อน

Pinout & Configuration



Clock 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

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

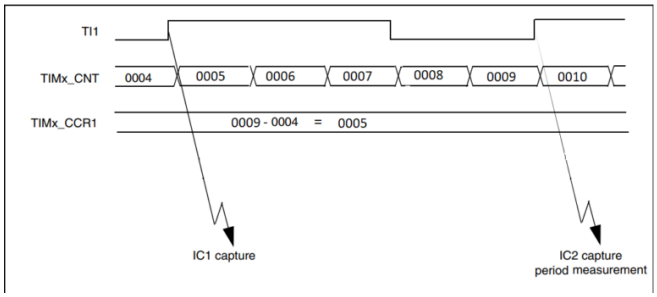
S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

- 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

$$\boldsymbol{R} = 97.859 * ln(\boldsymbol{f_R}) - 770.34$$
$$\boldsymbol{G} = 89.49 * ln(\boldsymbol{f_G}) - 666.01$$
$$\boldsymbol{B} = 102.26 * ln(\boldsymbol{f_B}) - 823.62$$

$\boldsymbol{R}, \boldsymbol{G}, \boldsymbol{B}$ คือ ค่า RGB ของสีแดง เขียว น้ำเงินตามลำดับ มีค่าตั้งแต่ 0 – 255 (หากคำนวณได้น้อยกว่า 0 หรือมากกว่า 255 ให้ เปลี่ยนเป็น 0 หรือ 255)

$\boldsymbol{f_R}, \boldsymbol{f_G}, \boldsymbol{f_B}$ คือ ค่าความถี่ที่วัดได้ของสีแดง เขียว น้ำเงินตามลำดับ

- ในการวัดสีของสิ่งของ จะนำค่า R G B ที่ได้จากการวัดอย่างละ 100 ครั้ง ไปคิดค่าเฉลี่ย เพื่อให้ได้ค่าที่แม่นยำยิ่งขึ้น

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	

Sensor calibration

1. เก็บข้อมูลความถี่ที่วัดได้จาก sensor และ ค่า RGB ของสิ่งของที่วัด
   - ใช้แอพ ColorPicker เพื่อหาค่า RGB ของสิ่งของ
   - ใช้เว็บไซต์ https://www.w3schools.com/colors/colors_rgb.asp เพื่อตรวจสอบและปรับสีที่ได้จากแอพให้ตรงกับความจริง
   - เก็บข้อมูลทั้งหมด 100 ครั้ง สามารถอ่านเพิ่มได้ในไฟล์ sensor_calibrate.xlsm

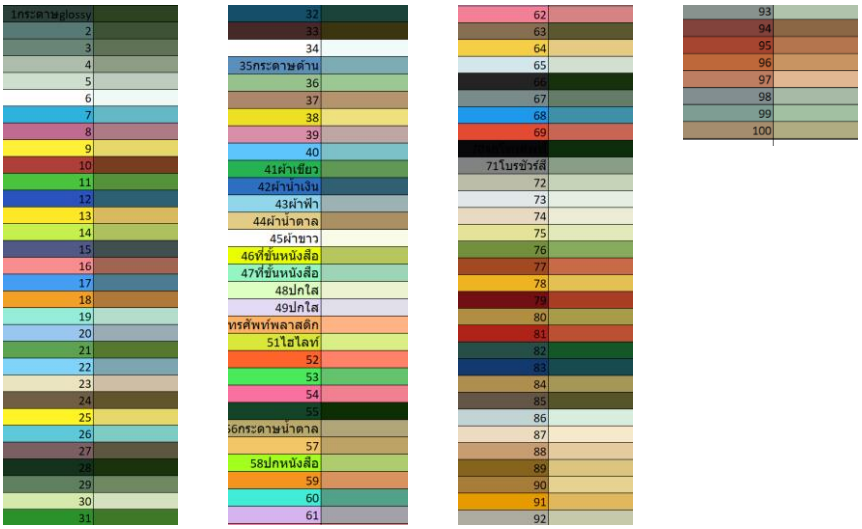
Real color	Frequency	Real R	Frequency	Real G	Frequency	Real B
1กระดาษglossy	4104	43	3567	64	4336	56
2	4965	87	4313	122	5184	119
3	6891	105	6076	133	7291	120
4	11404	173	9820	191	11706	172
5	18268	206	16555	223	20420	207
6	30179	255	27761	255	35353	255
7	7351	46	13329	179	21580	222
8	15135	192	6746	107	11525	144
9	27892	255	19170	241	9066	55
10	8844	173	3414	64	4299	58
11	6271	74	8679	196	5505	63
12	4183	42	4979	83	9830	189
13	24251	255	13649	232	7886	40
14	15717	198	14744	241	7883	76
15	5029	81	4188	90	6742	134
16	13710	248	5296	142	7058	144
17	5630	68	6801	158	13001	244
18	15783	242	6600	160	5470	38
19	16737	149	20123	238	23389	218
20	12650	154	11725	199	18540	239
21	6230	93	6585	163	4990	81
22	9407	128	11023	211	17578	249
23	21249	235	14247	228	15738	193

2. นำความถี่ และ ค่า RGB มาสร้างกราฟ และหาสมการในการแปลงความถี่เป็น RGB ด้วย คำสั่ง Trendline ของโปรแกรม Excel

3. พบว่า logarithmic equation มีแนวโน้มของกราฟใกล้เคียงกับข้อมูลมากที่สุด จึงเลือกใช้
4. สามารถปรับค่าสัมประสิทธิ์ของ logarithmic equation ที่ #define ใน main.c

```
//value for sensor calibration
#define RED_a 97.859
#define RED_b -770.34
#define GREEN_a 89.49
#define GREEN_b -666.01
#define BLUE_a 102.26
#define BLUE_b -823.62
```

5. ส่วนใหญ่ได้ผลลัพธ์ใกล้เคียงสีของสิ่งของ 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

จากข้อมูลในเว็บไซต์ด้านบนและจากการวิเคราะห์ข้อมูลค่าสีพบว่าควรตรวจสอบโดย

ถ้า

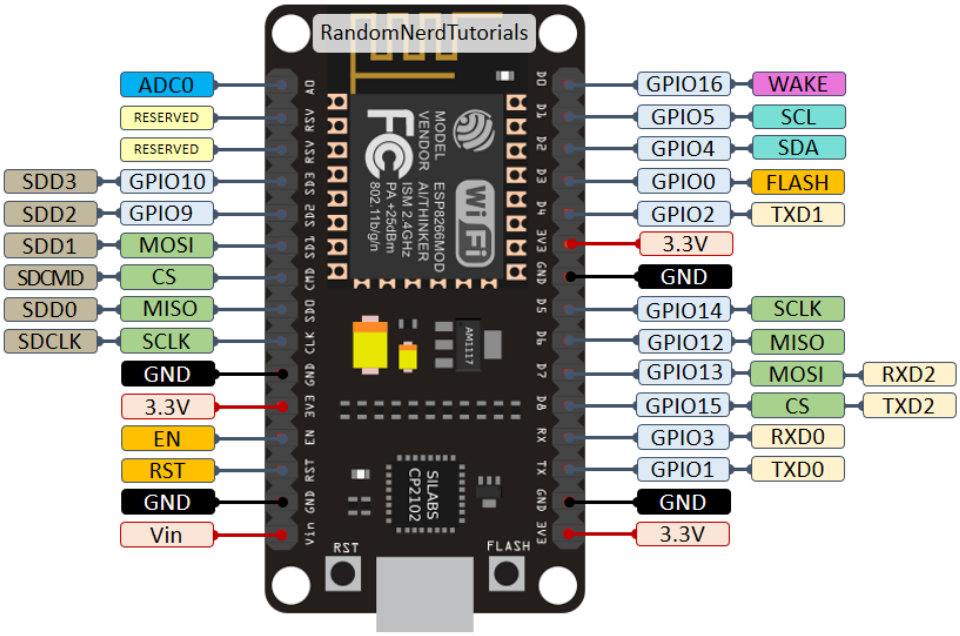
1. (ผลต่างค่า R)^2 + (ผลต่างค่า G)^2 + (ผลต่างค่า B)^2 <= 7000 หรือ
2. (ผลต่างค่า Hue <= 40 หรือ >= 340) และ ผลต่างค่า Saturation <= 70 และ ผลต่างค่า Lightness <= 40

แสดงว่าสีคล้ายกัน

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 เมื่อรับข้อมูลมาจนเจอ ‘\n’ จะทำการส่งข้อมูลต่อไปให้ Firebase ผ่าน Wi-Fi ที่กำหนดชื่อและรหัสไว้แล้ว

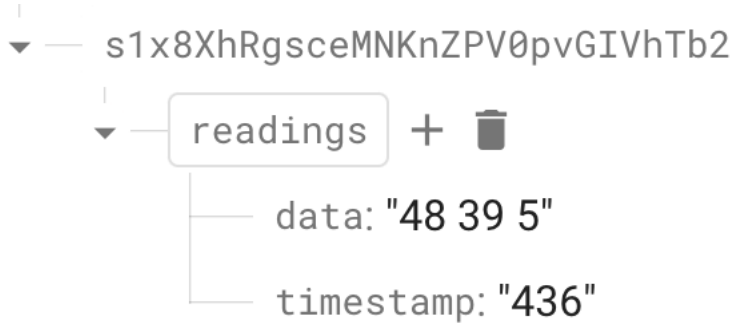
วิธีการต่อเข้ากับ STM32

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 โดย data เก็บข้อมูลสีที่บอร์ดอ่านได้ timestamp เก็บเลขจำนวนครั้งที่สแกนตั้งแต่รีเซ็ทบอร์ด โดยใช้เพื่อตรวจสอบว่าข้อมูลที่เข้ามาใหม่เป็นข้อมูลใหม่หรือข้อมูลเก่า

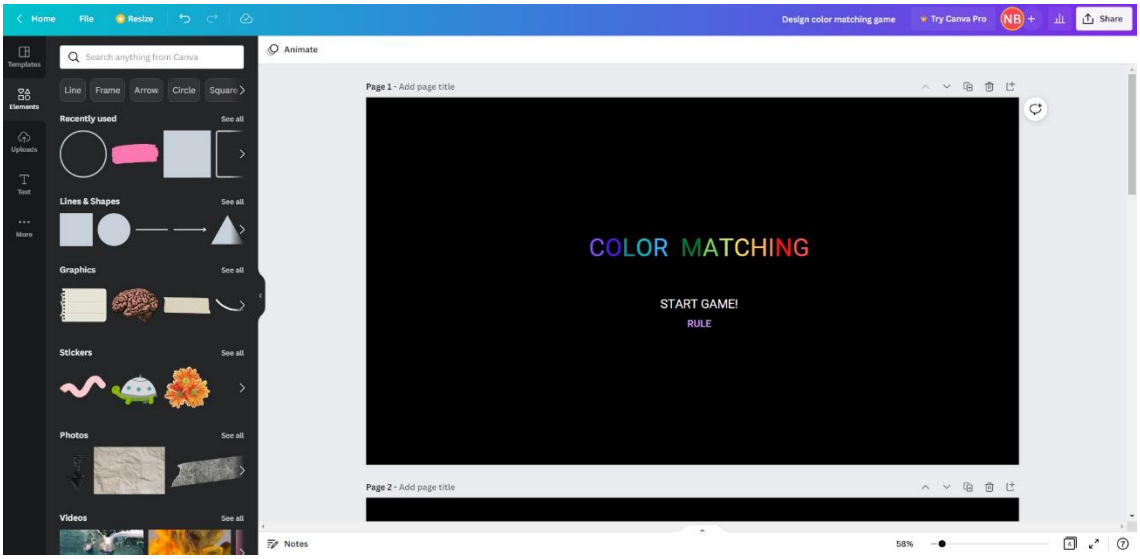
เมื่อข้อมูลถูกส่งมาจาก NodeMCU ข้อมูลจะถูกจัดเก็บใน database และ website จะดึงข้อมูลจาก firebase ไปแสดงผล

<u>แหล่งที่มา</u>

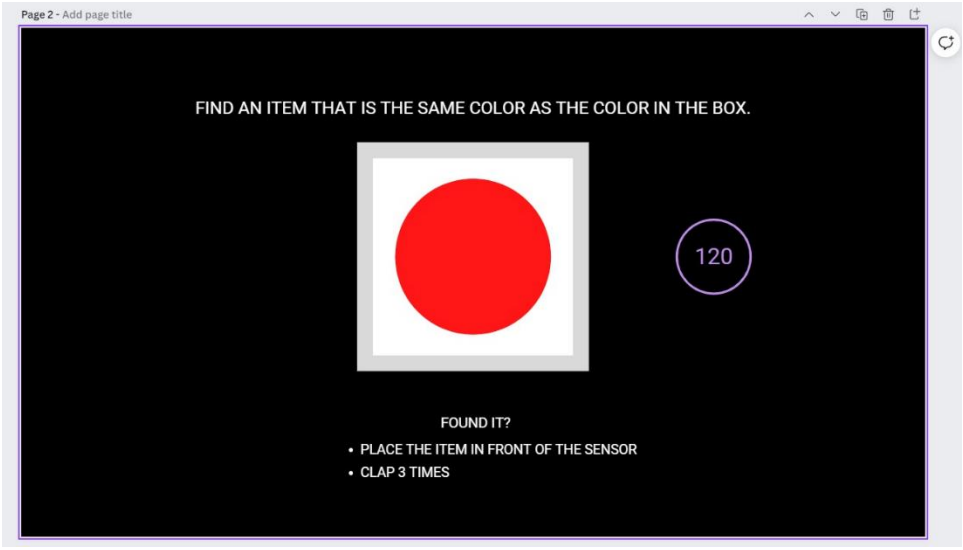
- https://randomnerdtutorials.com/esp8266-data-logging-firebase-realtime-database/

UX/UI

ออกแบบ Theme เกมที่ต้องการลงในเว็บไซต์ https://www.canva.com/



รูปภาพหน้าเริ่มเกมของ website



รูปภาพหน้าระหว่างเล่นเกมหลังกดปุ่ม START GAME

Website

พัฒนา website ตามที่ฝ่าย UX/UI ที่ออกแบบมาโดยใช้ React และ Material UI ในการเขียนหน้า website และดึงข้อมูลจาก firebase ซึ่งเป็นข้อมูลตัวเลขระบุค่า rgb ของสีที่อ่านได้จาก color sensor

นอกจากนี้ยังมีการพัฒนา logic ต่างๆในเกมเพิ่มเติม ได้แก่

- การเปรียบเทียบสี จะขึ้นโชว์สีของสิ่งของที่วัดได้ผ่าน color sensor เมื่อสีที่ผู้เล่นหามาได้นั้นไม่ใกล้เคียงกับตามที่เว็บไซต์กำหนด
- การสุ่มสีขึ้นใหม่เรื่อยๆ เพื่อให้ผู้เล่นได้หาสิ่งของที่สุ่มกำหนดให้จากหน้าเว็บไซต์
- deploy website หลังเขียนเกมตามที่ออกแบบเสร็จทั้งหมดบน netlify

Picture Credits

- 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
- https://www.section.io/engineering-education/how-to-implement-material-ui-in-react/
- https://medium.com/@panacholn/%E0%B8%A1%E0%B8%B2%E0%B8%A5%E0%B8%AD%E0%B8%87-deploy-next-js-project-%E0%B8%9A%E0%B8%99-netlify-%E0%B8%81%E0%B8%B1%E0%B8%99%E0%B9%80%E0%B8%96%E0%B8%AD%E0%B8%B0-e62b2465bcf5