อุปกรณ์ตรวจจับระยะห่างระหว่างคิวโรงอาหาร
และ จำกัดจำนวนคิวซื้ออาหาร

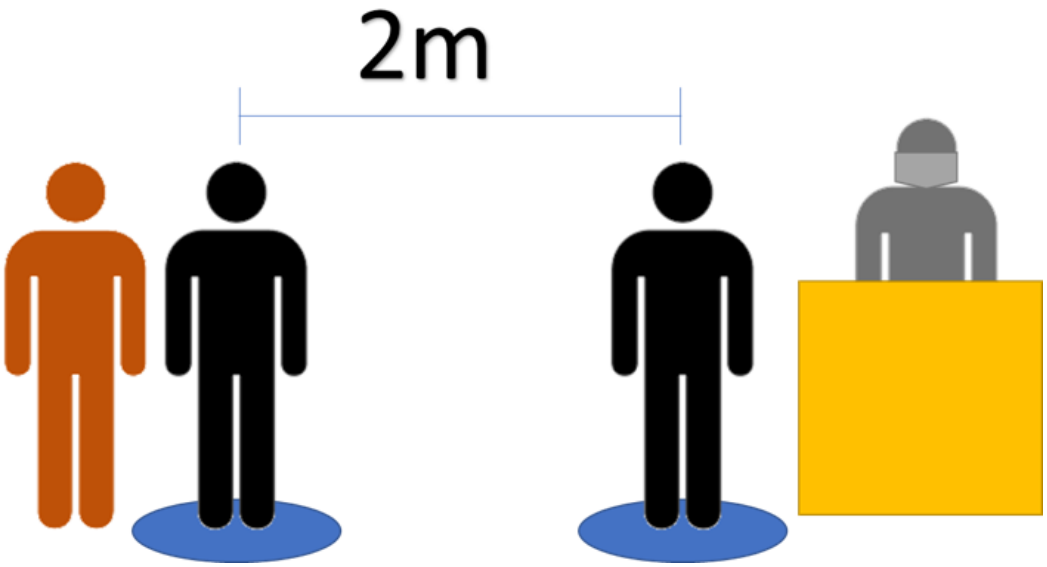
โดย

ธฤต แสงสุวรรณ

ธัญธร ฐิติภัทรยรรยง

ธนวินท์ ทีวะเวช

ชินกฤต เอกวณิชสกุลพร



รายงานนี้เป็นส่วนหนึ่งของวิชา

2110366 Embedded System Laboratory

เทอม 2 ปีการศึกษา 2563 จุฬาลงกรณ์มหาวิทยาลัย

บทนำ

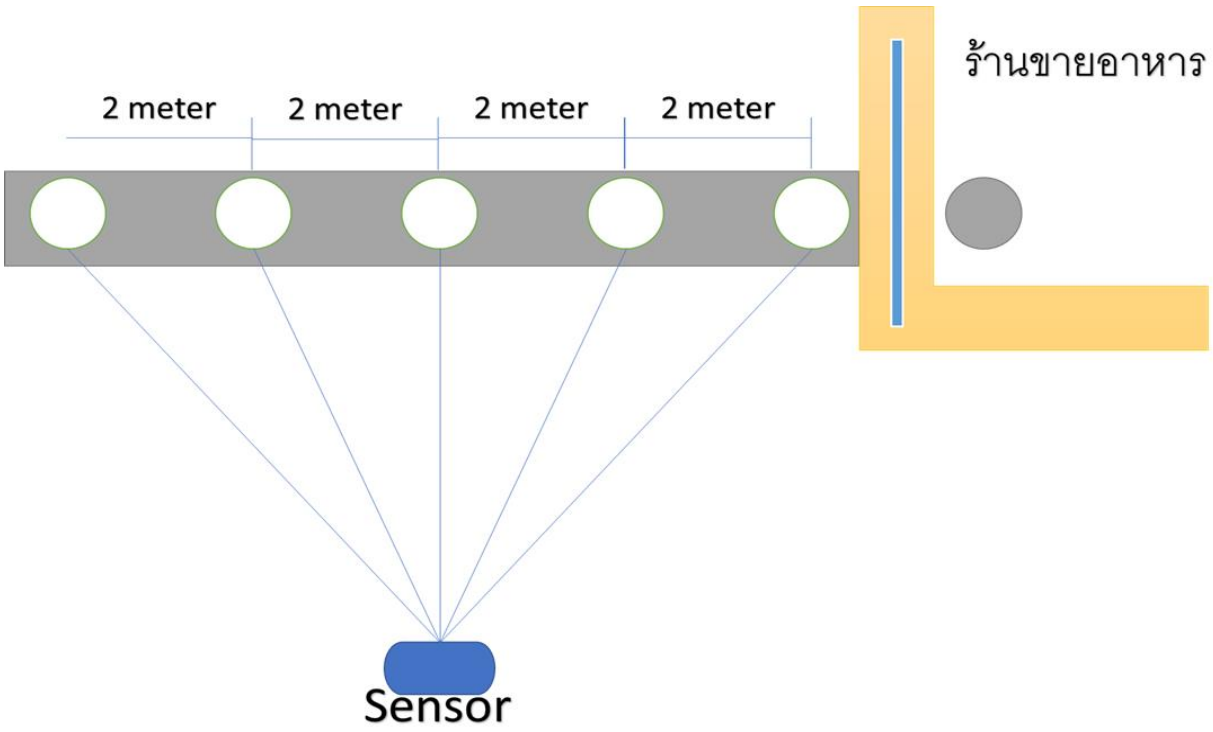
ในสถานการณ์ Covid-19 ในปัจจุบันนั้น ทำให้ผู้คนต้องระวังตัว และ รักษาความสะอาดกันมากขึ้น รวมถึงต้องปรับเปลี่ยนวิถีชีวิต เช่น ทำงานที่บ้าน ล้างมือบ่อยขึ้น ใส่หน้ากาก อยู่ห่างกันมากขึ้น ฯลฯ แต่ถึงแม้ว่าจะปรับเปลี่ยนวิถีชีวิตยังไง ก็มีบางคน หรือบางสถานการณ์ ที่ไม่สามารถหลีกเลี่ยงที่จะทำได้ เช่น การที่ต้องไปต่อแถวซื้ออาหาร ที่มีความเสี่ยงต่อการติดเชื่อ เป็นต้น

ทางกลุ่มของเรานั้นได้ตระหนัก และเล็งเห็นถึงความสำคัญของ การเว้นระยะห่างเพื่อความปลอดภัย ทางเราจึงมีความเห็นที่จะทำ เครื่องที่ใช้ในการ วัดระยะห่างระหว่างคิว ต่อแถวซื้ออาหาร และ นับจำนวนคนที่จะมาซื้ออาหาร และ จำกัดคิวการซื้ออาหาร เพื่อลดโอกาสในการแพร่เชื้อ

เกี่ยวกับ Project นี้

กลุ่มของพวกเรามีแนวคิดที่จะทำ เครื่องที่ไว้สำหรับเช็คระยะห่างของคนที่มาต่อแถวซื้ออาหาร เนื่องจากเรามองว่า การเข้าแถวต่อคิวซื้ออาหารนั้นเป็นหนึ่งในกิจกรรมในชีวิตประจำวันที่ทำให้เรามีโอกาสสัมผัสกับผู้อื่นมาก และความเสี่ยงในการแพร่เชื้อให้ผู้อื่นสูงอีกด้วย

นอกจากนี้ ทางเรามีความคิดว่าจะติดตั้งปุ่มสำหรับแม่ค้าเพื่อนับจำนวนคนที่มาซื้ออาหาร เก็บไว้เป็นข้อมูลทางสถิติ ให้ผู้คนพิจารณาการซื้ออาหาร ของร้านนั้นๆด้วย



อุปกรณ์ที่ใช้

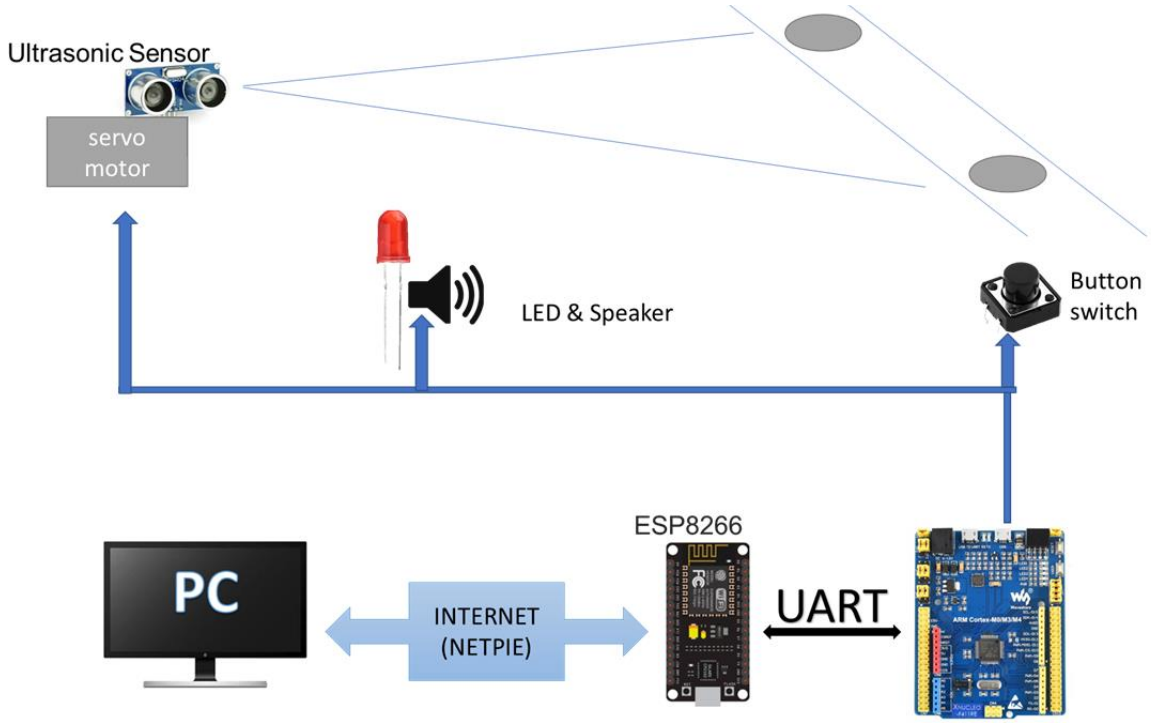
1. NUCLEO-F411RE
2. Ultrasonic Sensor
3 .Servo Motor
4. Speaker
5 .Photo Interrupter (ITR9608)
6. ESP8266 (wifi module)
7. Battery เลี้ยง esp8266
8. LED
9. Button Switch

รายละเอียดอุปกรณ์

สำหรับรายระเอียดตัวงานของพวกเรา จะมี

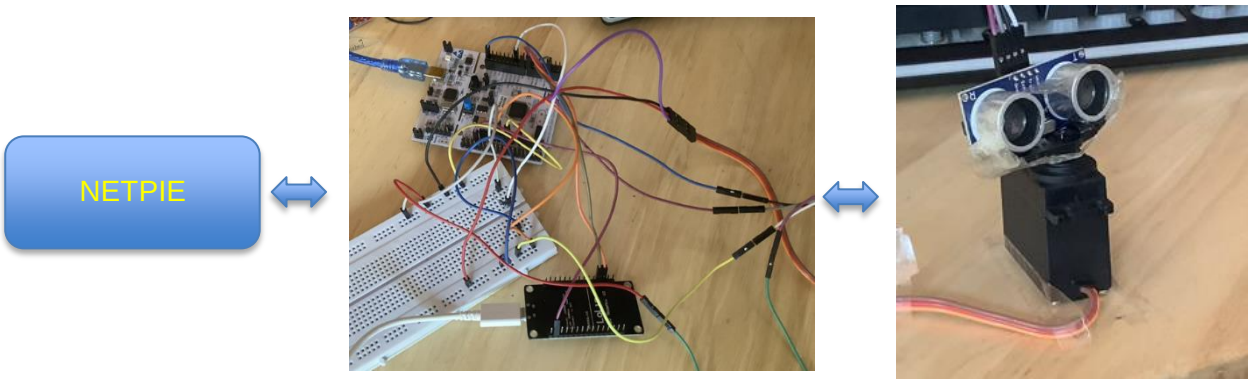
- บอร์ด NUCLEO-F411RE ไว้สำหรับสั่งงาน อุปกรณ์แต่ละตัว
- Servo motor และ Ultrasonic Sensor สำหรับ
  ตรวจว่าคนแต่ละคนอยู่ในตำแหน่งของตัวเองหรือไม่ โดย Servo Motor จะหมุนตัว Ultrasonic Sensor
  ให้ชี้ไปที่ตำแหน่งที่ยืนรอคิวเพื่อดูว่าตำแหน่งรอคิวแต่ละตำแหน่งมีคนไหม แล้วมีคนยืนผอดตำแหน่งหรือไม่
- LED จะสว่างเมื่อจับได้ว่ามีคนอยู่ชิดกันเกินไป
- Speaker จะส่งเสียงเมื่อจับได้ว่ามีคนอยู่ชิดกันเกินไป
- Button Switch ไว้สำหรับแม่ค้าไว้นับจำนวนคนที่มาซื้อ
- ESP8266 ใช้รับข้อมูลจาก NUCLEO อาทิเช่น จำนวนคนที่มาซื้ออาหาร หรือแจ้งเตือนคนอยู่ใกล้กันเกินไปในแถวคิว เป็นต้น ส่งไปยัง NETPIE

แผนภาพจำลองการเชื่อมต่อของแต่ละอุปกรณ์

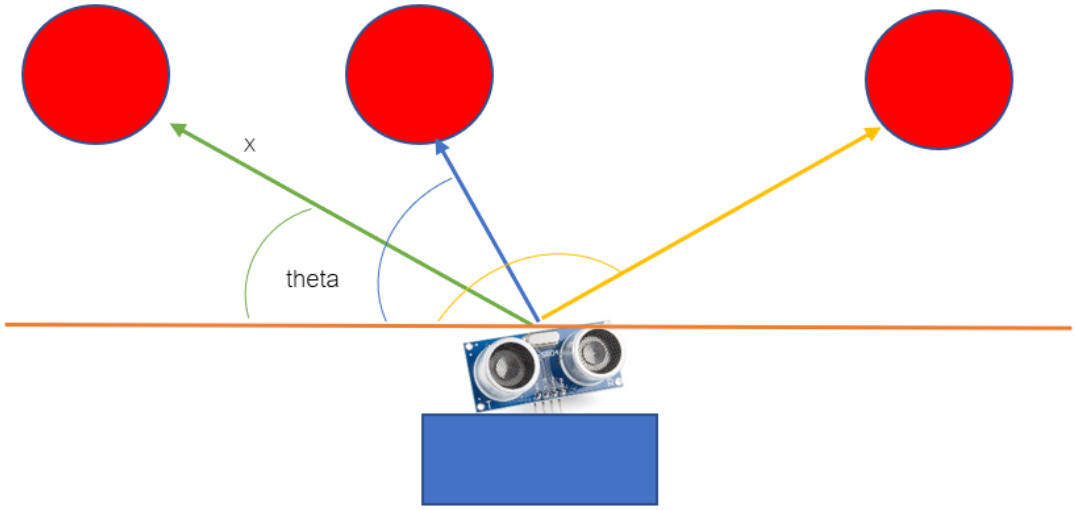


วิธีใช้งาน

เราจะตั้ง **Ultrasonic Sensor** ที่ติด **Servo motor** ตั้งไว้ห่างจากแถว เพื่อให้ **Sensor** ตรวจสอบ ว่ามีคนอยู่ในแถวอยู่ชิดกัน หรือไม่ หากมีคนอยู่ติดกันเกินไป จะมี **LED** แจ้งเตือนให้คนในแถวเห็นว่าอยู่ชิดกันเกินไป แล้ว จะทำการส่งข้อมูลไปเก็บไว้ในหน้า **website:** **https://embed112.web.app/** ให้สามารถตรวจสอบย้อนหลังได้ **1** วัน และ เรายังมีปุ่มที่อยู่กับทางฝั่งแม่ค้า ให้กดเพื่อส่งจำนวนคนที่มาต่อแถว แล้วเก็บข้อมูลลงเว็บ โดยแนบเวลาไว้ เช่น คิวที่เท่าไหร่ มาเวลาไหน เป็นต้น



Algorithm วัดระยะห่างของคนในคิว



สำหรับส่วน มีไว้ใช้เพื่อหาว่าในขณะที่คนกำลังยืนต่อแถวนั้น มีคนไหนยืนอยู่ใกล้กันเกินสองเมตรหรือไม่ โดยจากรูป ให้วงกลมสีแดง แทนคนที่กำลังยืนต่อคิวอยู่ หลักการทำงานก็คือตัว servo motor จะหมุน ultrasonic sensor ไปเป็นวงรอบบริเวณที่มีคนกำลังเข้าคิวอยู่ จากนั้นเมื่อพบคนในคิว ก็วัดระยะระหว่าง ultrasonic กับคนที่พบในคิว (เช่น X) และก็วัดมุมที่ servo motor หมุนมา (เช่น theta) เทียบกับเส้นขนานสีแดง จากนั้นจึงคำนวณระยะทางในแนวขนานกับเส้นสีแดงผ่านสูตร X*cos(theta)

```
float find_X_Distance(float dis , int angle){
    return dis * cos( (float)angle / (float)180 * PII  );
}
```

เมื่อได้ระยะทางในแนวขนานมาแล้วก็เก็บไว้เป็น last_x_dis แล้วทำการคำนวณระยะทางของคนต่อไปด้วยวิธีการแบบเดียวกัน เก็บไว้เป็น x_dis จากนั้นก็นำมาหาผลต่างระยะห่างระหว่างคนทั้งสอง ด้วยวิธีการหา last_x_dis – x_dis หากทั้งสองคนอยู่ใกล้กันเกินสองเมตร แปลว่าคนสองคนนั้นอยู่ใกล้กันเกินระยะห่างปลอดภัยตามมาตรการ social distancing แล้ว ก็ให้อุปกรณ์ทำการเปิดไฟแจ้งเตือนว่ามีคนอยู่ใกล้กันเกินสองเมตร แล้วก็ให้ส่งข้อมูลเวลาที่มีคนอยู่ใกล้กันเกินไปเพื่อนำไปบันทึกประวัติ และแสดงผลผ่านทางเว็บไซต์

NUCLEO-F411RE ALGORITHM DISTANCE CODE

```
dis_notice_blink_down();
last_x_dis = 10000;
uint8_t should_bring_down = 1;
clear_tracking();
set_angle(60);
for (int angle = 80; angle <= 150; angle++){
    set_angle(angle);
    float dis = find_Distance_stable();
    uint8_t pre_save = found_people(dis, angle);
    if ( (pre_save == 1 ) && (tracking[angle-1] == 0) ){
        float x_dis = find_X_Distance(dis, angle);
        float distancing = last_x_dis - x_dis;
        last_x_dis  = x_dis;
        tracking[angle] = 1;
        if (distancing < 50){
            dis_notice_blink_up();
            should_bring_down = 0;
        }
    }else{
        tracking[angle] = pre_save;
    }
}
if (should_bring_down == 1){ dis_notice_blink_down(); }
```

กรณี คนไกล้กันเกินไป

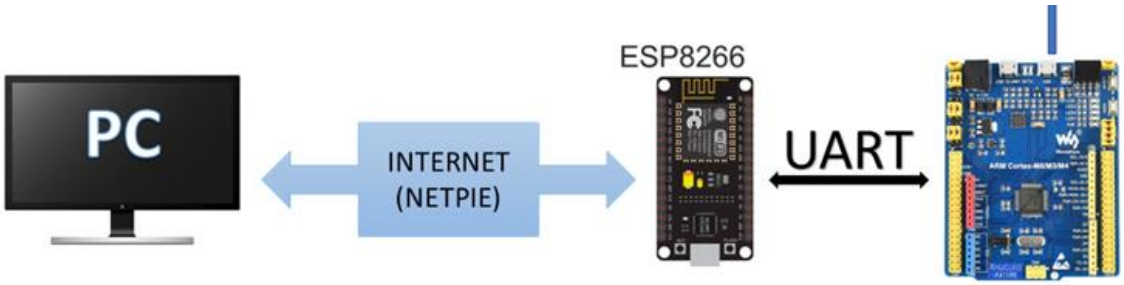
QUEUE CHECKER

ร้านข้าวแกงลุงพล

Current Queue = 9 / 7

Distance Status = Hazard

การส่งข้อมูลระหว่าง NUCLEO-F411RE ESP8266 **และ** NETPIE



NUCLEO-F411RE Code ในส่วนการส่งข้อมูลไปยัง ESP8266

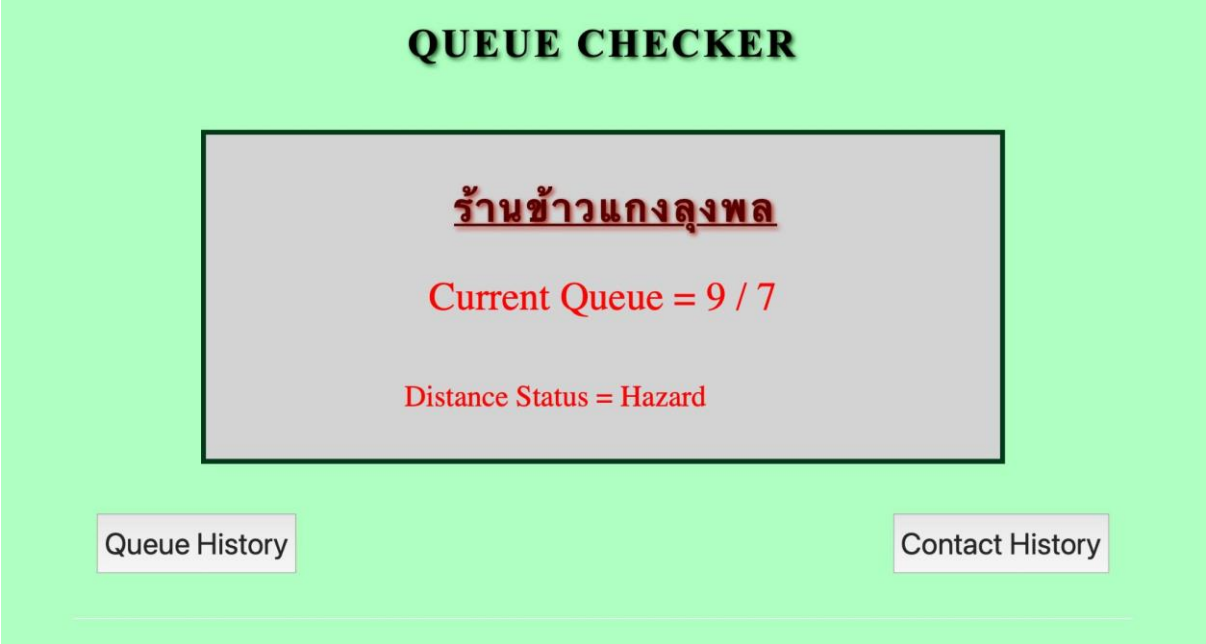
```
void i2csenter(){ //เนื่องจาก I2C ไม่สามารถใช้กับ ESP8266 จึงเปลี่ยนมาใช้ UART แทน
    char sentData[] = { 48+cur, 48+cls, 65};
    if ( HAL_UART_Transmit(&huart1, sentData, 3, HAL_MAX_DELAY) == HAL_OK){
        int cxccgbfgfg = 0 ;
    }else{

    }

}
```

ESP8266 รับข้อมูลมาแล้วแปลงรูปแบบเป็น **JSON** แล้วส่งไปให้ **NETPIE** ต่อเพื่อเอาไปแสดงในหน้าเว็บ โดย ข้อมูลที่ **NETPIE** ดึงไป นั้น เราจะเขียน ในหัวข้อถัดไปว่าจะทำอย่างไรกับข้อมูล

```
if(NodeSerial.available()>=3){
   Serial.println("rec");
   Serial.println(NodeSerial.available());
   char currentQ = NodeSerial.read();
   char contact = NodeSerial.read();
   while(NodeSerial.available()> 0) NodeSerial.read();
   String jsonData;
   String data = "/" + String(currentQ) + "/" +String(contact) + "/";
   char msg[128];
   data.toCharArray(msg,data.length());
   Serial.println(msg);
   microgear.chat(TargetWeb , msg);
   jsonData = "{\"contact\":";
   jsonData += String(contact);
   jsonData += ",\"currentQ\":";
   jsonData += String(currentQ);
   jsonData += "}";
   microgear.writeFeed(FEEDID,jsonData,"vieXGnCnRE2cRdelUBoRgIX5Qy2TdVgt");
    }
delay(10);
```

UI Designer and Development

ตัวเว็บที่ แสดงผลข้อมูลที่ประมวลผล มาแสดงใน **web application:**https://embed112.web.app/ โดยดึงข้อมูลจาก **netpie feed** ด้วย **Restful API** เพื่อแสดงตารางประวัติการเข้าคิวและประวัติการรักษาระยะห่างที่ไม่เพียงพอ และแสดงผลข้อมูลแบบ **real time** ผ่าน **library microgear (netpie)** ซึ่งส่งมาจาก **node mcu8266** ที่รับข้อมูล **sensor** จาก **stm32** อีกที



รูปภาพแสดงตัวอย่างหน้าเว็บ

web application (HTML + CSS + JAVASCRIPT)

ตัว web application นี้มีจะเป็น Code แบบ Single Web Page หรือก็คือมีการ รวม Code ในส่วนของ HTML , CSS และ JAVASCRIPT ในไฟล์เดียวกันหมด (index.html)

Code ของ CSS จะเก็บอยู่ใน tag <style> โดยจะเก็บข้อมูล สีและ ตกแต่งหน้าเว็บของเรา โดยเราจะไม่ขอลงละเอียดในส่วนนี้ (รูปภาพตัวอย่าง Tag <style> ส่วนหนึ่งของ Code)

```
<style>
  body {
      background-color: #aeffc2;
  }
      .div1 {
          position: relative;
          background-color: lightgrey;
          width: 600px;
          border: 5px solid rgb(1, 58, 25);
          padding: 100px;
          margin: auto;
      }
      .red{
          color: red;
```

Code ในส่วนของ JavaScript หรือ JS จะอยู่ใน tag <script> โดยมีคำสั่งที่สำคัญมีดังนี้

1. Function การดึงข้อมูลที่ส่งมาจาก MCU8266

```
microgear.on('message',function(topic,msg) {
  var date = Date();
  var status;
  var split_msg = msg.split("/");
  console.log(msg);  // for debug
  if(typeof(split_msg[0])!='undefined' && split_msg[0]==""){

    document.getElementById("currentQ").innerHTML = "Current Queue = " + split_msg[1] + " / 7 ";
    var a = parseInt(split_msg[1]);
    if(a >= 5){
      document.getElementById("currentQ").classList ="var2r";
    }
    else if(a >= 3){
      document.getElementById("currentQ").classList ="var2o";
    }
      else{
      document.getElementById("currentQ").classList = "var2g";
    }
    if(split_msg[2] == "1"){
        status = " Hazard";
        document.getElementById("distance").innerHTML = "Distance Status = " + status;
        document.getElementById("distance").classList = "var3r";
    }
    else{
        status = "OK Distance";
        document.getElementById("distance").innerHTML = "Distance Status = " + status;
        document.getElementById("distance").classList = "var3g";
    }
  }
});
```

เราจะใช้ ฟังก์ชั่น microgear.chat() เพื่อรับ data แบบ Real Time ของ Sensor จาก node mcu 8266 โดยมี NETPIE เป็น cloud platform ตัวกลางในการรับส่งข้อมูล จากนั้น เราจะนำ Data ที่ได้ โดยเป็นข้อมูลแบบ String มาแบ่งข้อมูล ออกเป็น 2 ชุด ด้วยกัน

1. currentQ บอกว่าปัจจุบันในคิวมีคนเข้าเท่าไหร่
2. contact บอกว่าขณะปัจจุบันมีคนไม่รักษาระยะห่างไหม

โดยจะนำข้อมูลที่ได้ มาแสดงใน HTML โดยข้างในมีฟังก์ชั่นที่เปลี่ยน class css ของ tag ที่ข้อมูลนั้นอยู่ ตามแค่ว่าข้อมูลนั้นมากน้อยเท่าใด ด้วย

2. Function ของปุ่มขอดูข้อมูล

```
function d(mode) {
  var request = new XMLHttpRequest();
  request.open('GET', 'https://api.netpie.io/feed/embed112Feed?apikey=vieXGnCnRE2cRdelUBoRgIX5Qy2TdVgt&granularity=10seconds&since=1day&filter=currentQ,contact', true);
  request.onload = function () {
// Begin accessing JSON data here
      var myjson = JSON.parse(this.responseText);
      console.log(myjson);
      console.log(request.status);
      if(request.status >= 20 && request.status <400){
          console.log(myjson);
          if(mode == 1){
              document.getElementById("tab").innerHTML = genContactTable(myjson);
          }else{
              document.getElementById("tab").innerHTML = genQTable(myjson);
          }
        }
      }
    request.send(null);
}
```

เมื่อมีคนกด **button Queue History** หรือ **Contact History** จะส่งค่า **parameter mode** แล้วแต่ว่าเป็นปุ่มไหน โดยมันจะไปใช้ **GET method** ของ **Restful API** เพื่อ **request** ข้อมูลแบบ **json** จาก **feed NETPIE** ที่เราตั้งและเก็บค่าที่ส่งจาก **8266** ไว้ เมื่อได้ข้อมูลแล้ว ก็จะส่งข้อมูลไปทำใน ฟังก์ชั่น **genContactTable** หรือ **genlQtable** ตาม **mode** ที่เราได้รับมา ฟังก์ชั่น **gen** เหล่านี้จะให้ตารางออกมา ซึ่งจะถูกนำไปใส่ไว้ใน **tag table** ส่วน **html** ต่อไป

Function genContactTable

```
function genContactTable(myjson){
  var header = '';
  var valueRows = '';
  header += '<tr><th><center>Contact Time</th></tr>';
  for(i=0;i<myjson.data[0].values.length;i++){
          if(myjson.data[0].values[i][1] == '1'){
          var date = myjson.data[0].values[i][0];
          var s1   = new Date(date).toLocaleDateString("th-TH");
          var s2 = new Date(date).toLocaleTimeString("th-TH");
          var rdate = s1 + '   ' + s2;
          var value = '<tr><td><center>' + rdate + '</center></td></tr>';
          valueRows += value;
      }
  }
  table = header + valueRows;
  return table;
  }
```

Function genlQtable

```
function genQTable(myjson){
  var header = '';
  var valueRow = '';
  var valueRows = '';
  var value = '';
  var table = '';
  var j = 0;
  header += '<colgroup><col style="width: 33%"/><col style="width: 67%"/>'
    + '</colgroup><tr><th><center>Queue Number</th><th><center>Time</th></tr>';
  if(myjson.data[1].values.length>0){
    console.log("ok");
  var oldQ = parseInt(myjson.data[1].values[0][1])-1;
  for(i=0;i<myjson.data[1].values.length;i++){
    console.log("ok1");
      value = '';
      valueRow = '';
      if(parseInt(myjson.data[1].values[i][1]) > oldQ){
          console.log("ok2");
          var date = myjson.data[1].values[i][0];
          var s1  = new Date(date).toLocaleDateString("th-TH");
          var s2 = new Date(date).toLocaleTimeString("th-TH");
          var rDate = s1 + '   ' + s2;
          j += 1;
          value += '<td><center>' + String(j) + '</td></center>';
          value += '<td><center>' + (rDate) + '</td></center>';
          valueRow = '<tr>' + value +'</tr>';
          valueRows += valueRow;
       }
       oldQ = parseInt(myjson.data[1].values[i][1]);
      }
  }
  table = header + valueRows;
  return table;
  }
```

ตัวอย่างตารางแสดงข้อมูลที่ได้

Queue History

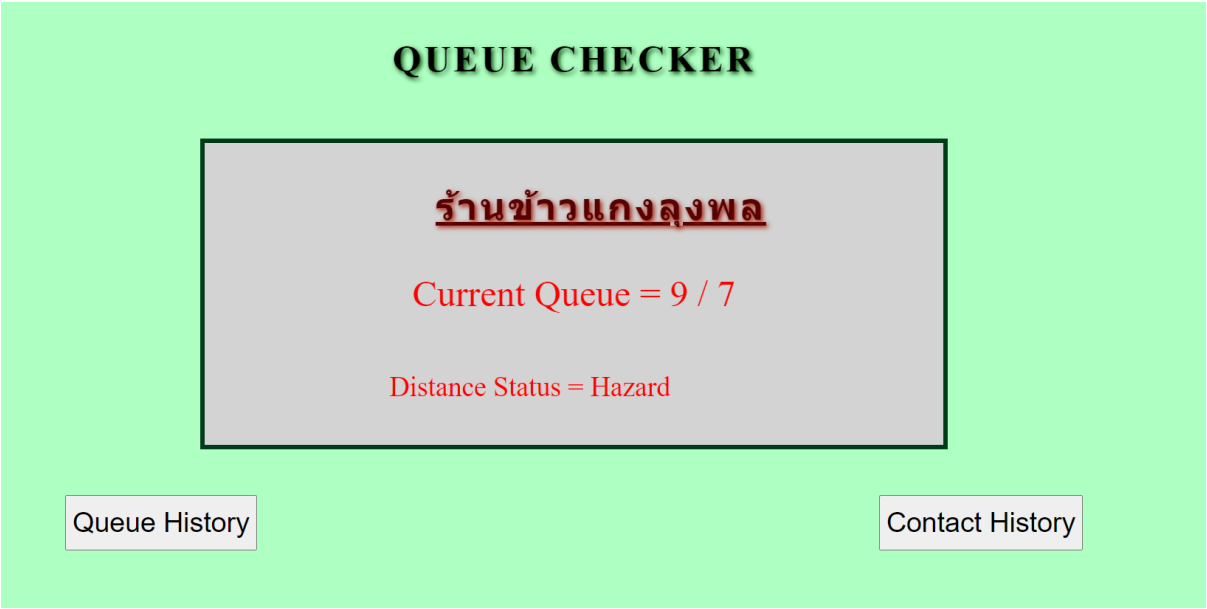
Contact History

Queue Number	Time
1	2/6/2564 07:36:36
2	2/6/2564 07:36:46
3	2/6/2564 07:36:56
4	2/6/2564 07:37:06
5	2/6/2564 07:38:38
6	2/6/2564 07:38:52
7	2/6/2564 08:22:36
8	2/6/2564 08:30:04
9	2/6/2564 08:34:18
10	2/6/2564 08:34:55
11	2/6/2564 08:35:07
12	2/6/2564 08:37:28
13	2/6/2564 08:44:42
14	2/6/2564 08:45:37
15	2/6/2564 08:47:16
16	2/6/2564 09:57:21
17	2/6/2564 09:57:33
18	2/6/2564 09:58:38
19	2/6/2564 09:58:55

Code ในส่วนสุดท้ายก็จะเป็นในส่วน HTML ที่อยู่ใน tag <body> ที่จะแสดงออกมาเป็นตัวหน้าเว็บนั่นเอง

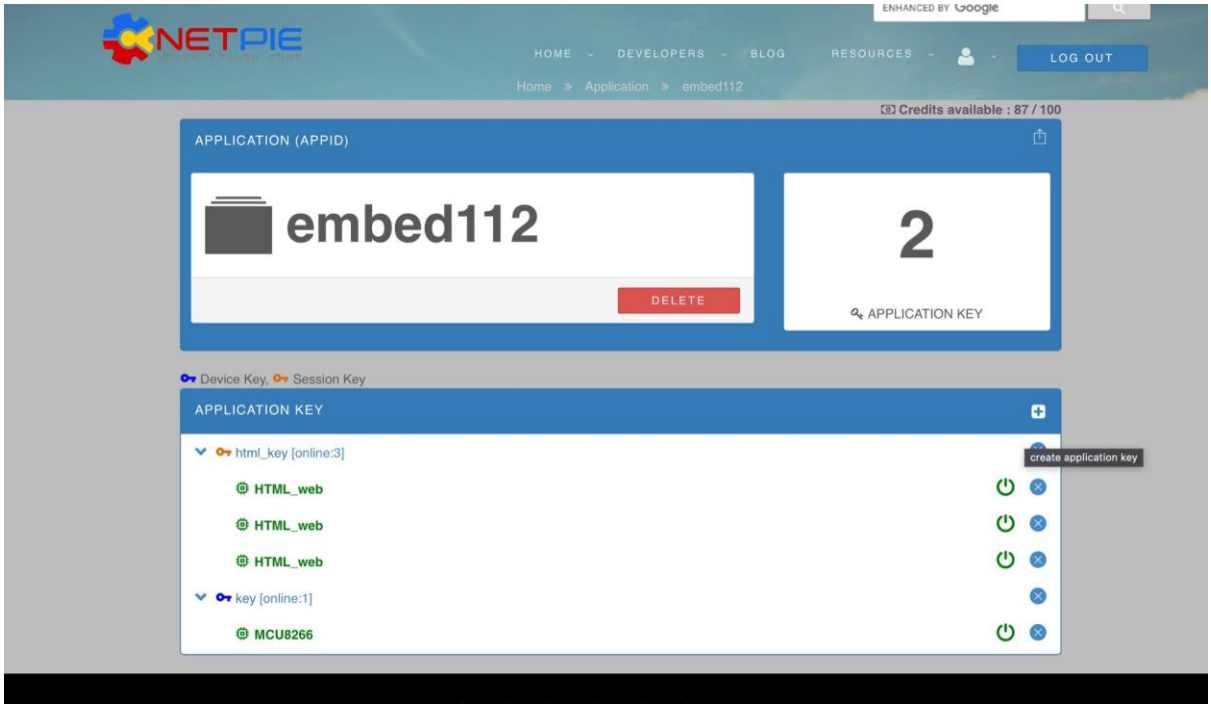
```
<h1 id="connected_NETPIE" ></h1>
<div class = "div1"> <h2 id="gg">ร้านข้าวแกงลุงพล</h2>
                    <p class="var2g" id="currentQ">current queue</p>
                    <p class="var3g" id="distance">distance status</p></div>

<div class="div2">
<button  type="button" class="var4" onclick="d(2)">Queue History</button>
<button  type="button" class="var5" onclick="d(1)">Contact History</button>
<div class="sss"></div>
</div>
<table id="tab"></table>
<div class="ss"></div>
</body>
```

หน้าเว็บ



เมื่อสร้างไฟล์ index.html เสร็จแล้วก็นำไป host ไว้ยัง Firebase เพื่อที่คนอื่นๆที่เข้าถึง internet ได้สามารถเข้าถึงข้อมูลของ queue checker ของเราได้ด้วย

NETPIE



สร้าง application project ใน netpie จากนั้นสร้าง application key ทั้ง key ธรรมดาๆ สำหรับอุปกรณ์ embeded (8266) และ html_key สำหรับใช้บนเว็บ เพื่อที่จะให้สามารถเข้าถึง netpie cloud กลาง ทำให้สามารถรับส่งข้อมูลระหว่างเว็บและ 8266 ได้ผ่าน internet ไม่ต้องต่อสายใดๆ

สร้าง netpie feeds เพื่อเป็น cloud storage ไว้เก็บข้อมูลจาก 8266 ที่รับจาก sensor ต่างๆ ไว้ (รับจาก stm32 อีกที) ซึ่งจะบันทึกเวลาไว้ด้วย เหมาะสำหรับการนำมาสร้างเป็นตารางบนเว็บที่เรากล่าวไว้ข้างต้น เมื่อมี feeds ทำให้เว็บของเราเข้าถึงและดึงข้อมูลที่เก็บไว้ซึ่งไม่สามารถเก็บในอุปกรณ์ embeded ได้สะดวกมาก

หน้าที่ต่างๆของสมาชิก

ธฤต แสงสุวรรณ

role : System Architecture

- ออกแบบ Algorithm ต่างๆ ในตัวเครื่อง (หน้า 5)
- ช่วยหาข้อมูลวิธีการส่งข้อมูลระหว่าง ESP8266 กับ NETPIE

ธัญธร ฐิติภัทรยรรยง

role : Team Management

- รวบรวมข้อมูล รับผิดชอบรายงาน
- ช่วยหาข้อมูลวิธีการส่งข้อมูลระหว่าง ESP8266 กับ NETPIE

ธนวินท์ ทีวะเวช

role : Embedded System Development

- จัดการ Embedded system และ การเชื่อมต่อ บอร์ด (หน้า 7)
- ปรับเทียบ Algorithm กับตัวเครื่อง เพื่อให้ตัวเครื่องทำงานได้ถูกต้อง

ชินกฤต เอกวณิชสกุลพร

role : UI Designer and Development

- รับผิดชอบเรื่อง NETPIE, Firebase และ หน้า website (หน้า 8)
- ช่วยในเรื่อง การสื่อสารระหว่าง Front-end กับ Embedded system

GITHUB : https://github.com/Herobxxm/4HMOP_Embbed_Project