Smart Car Parking

จัดทำโดย

1. กัญญาภัค บุญยะภาส รหัสนักศึกษา 64010035 กลุ่ม 18

2. พิมลณัฐ ศรีเผด็จกุลชา รหัสนักศึกษา 64010605 กลุ่ม 19

3. ภาพพิชญ์ พงศ์พัฒนาวุฒิ รหัสนักศึกษา 64010670 กลุ่ม 19

4. สรวิชญ์ เลยวานิชย์เจริญ รหัสนักศึกษา 64010876 กลุ่ม 20

เสนอ

รศ.ดร. เจริญ วงษ์ชุ่มเย็น

โครงงานฉบับนี้เป็นส่วนหนึ่งของวิชา 01076006

Digital System Fundamentals

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 1 ปีการศึกษา 2565

บทนำ

โครงงานเรื่อง Smart Car Parking จัดทำขึ้นเพื่อจำลองการทำงานของที่จอดรถที่มีระบบอำนวยความสะดวกสบายแก่ผู้ใช้งานให้มากยิ่งขึ้น โดย Smart Car Parking มีกระบวนการทำงานคือ แสดงจำนวนที่จอดรถที่ยังว่างอยู่ มีไฟแสดงสถานะตามแต่ละตำแหน่งของที่จอดรถ และสามารถเปิด – ปิดไฟของแต่ละตำแหน่งที่จอดรถได้อย่างอิสระ พร้อมกับการจำลองการทำงานของไม้กั้นเปิด – ปิดโดยอัตโนมัติ

โดยรายงานนี้เป็นส่วนหนึ่งของวิชา Digital System Fundamentals รหัส 01076112 เพื่อที่อธิบายชิ้นงานตั้งแต่กระบวนการหาข้อมูล กระบวนการออกแบบ กระบวนการพัฒนา กระบวนการทดสอบ โดยประกอบไปด้วยข้อมูลทางเทคนิค วิธีการ เงื่อนไข แนวคิด เหตุผลของการออกแบบและพัฒนาชิ้นงานทั้งหมด หากรายงานนี้มีข้อมูลผิดพลาดประการใด ผู้จัดทำขออภัยไว้ใน ณ ที่นี้ด้วย

คณะผู้จัดทำ

13 ธันวาคม พ.ศ. 2565

กระบวนการหาข้อมูล

1. กำหนดหัวข้อ

ศึกษาหาข้อมูลจากสิ่งรอบตัวเมื่อศึกษาค้นคว้าแล้วพบว่ายังมีสถานที่บางพื้นที่ที่มีการจอดรถแบบแสดงจำนวนที่ว่างของที่จอดรถได้ไม่มากนัก เราจึงเริ่มกำหนดหัวข้อหลักให้เกี่ยวข้องกับที่จอดรถเพื่อที่จะนำมาตัดสินใจทำชิ้นงาน จนสรุปออกมาเป็นภาพและแก้ไขปัญหานี้ได้ จึงเลือกทำชิ้นงาน Smart Car Parking ขึ้นมา

2. ค้นหาข้อมูลและรวบรวมข้อมูล

เริ่มจากการหาหลักการทำงานของที่จอดรถตามสถานที่ต่างๆ จนมาถึงการทำงานของ sensor ทั้งทางที่กั้น sensor จับรถว่ามีรถจอดหรือไม่ จนถึงการวาดวงจรว่า การทำงานแต่ละอย่างวงจรควรจะเป็นไปในทางไหน

3. เลือกแหล่งข้อมูล

ศึกษาค้นคว้าหาจากแหล่งข้อมูลที่หลากหลายเพื่อให้ได้ความคิด ไอเดียที่มากยิ่งขึ้น และสามารถนำมาประยุกต์กันได้ และเลือกแหล่งข้อมูลที่มีความน่าเชื่อถือ แหล่งข้อมูลที่มีการทดสอบหรือลองทำจริงๆ มีตัวอย่างผลลัพธ์ให้ดู

4. เตรียมอุปกรณ์

อ้างอิงจากแหล่งข้อมูลที่ได้ศึกษามา และวิเคราะห์ออกมาว่าสิ่งที่จะทำ (โมเดลจำลองที่จอดรถ) ใช้อุปกรณ์หลัก คือ บอร์ด FPGA จำนวน 2 บอร์ด ในการแสดงผลจำนวนที่จอดรถที่ยังว่างอยู่ ไฟแสดงสถานะการจอดและสำหรับการควบคุมการเปิดปิดไฟ ในสถานที่จอดรถ

ลำดับ	อุปกรณ์	จำนวน	หมายเหตุ
1	Board FPGA	2	ควบคุมการทำงานและแสดงผล
2	IR Sensor	16	ใช้ในการควบคุมไม้กั้นประตูเข้า-ออก
3	Arduino Uno R3 + สาย USB	1	ควบคุมการทำงานจากคำสั่งของโปรแกรม
4	Servo	2	ใช้ในการควบคุมไม้กั้น
5	Breadboard		
6	สายจัมป์		
7	โมเดลรถของเล่น	8	ใช้ในการจำลองเป็นรถ
8	LED	8	ใช้เป็นไฟตามแต่ละช่องจอดรถ

5. การค้นหาและรวบรวมข้อมูล

สอบถามจากรุ่นพี่และศึกษาข้อมูลจากทางอินเทอร์เน็ต รวบรวมไฟล์ข้อมูลที่น่าสนใจที่สามารถนำมาประยุกต์ได้ไว้และแลกเปลี่ยนข้อมูลที่หามาได้กับเพื่อนในกลุ่มเพื่อสรุปและอ้างอิงในการทดลง

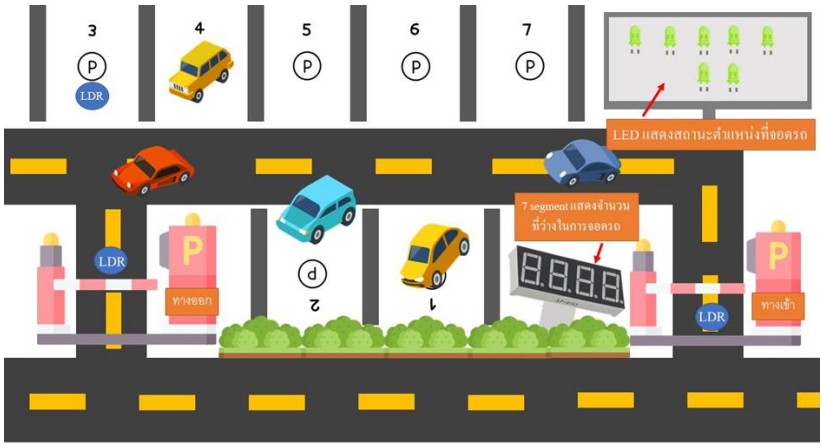
6. พิจารณาและสรุป

เมื่อค้นหาข้อมูลได้และทำตามทุกขั้นตอนเรียบร้อยแล้ว จึงทำการรวบรวมจดบันทึกข้อมูลที่สำคัญ นำข้อมูลที่ได้มาพิจารณาว่ามีความน่าเชื่อถือมากน้อยเพียงใดแล้ว และทำการสรุปเพื่อนำเสนอผลงาน

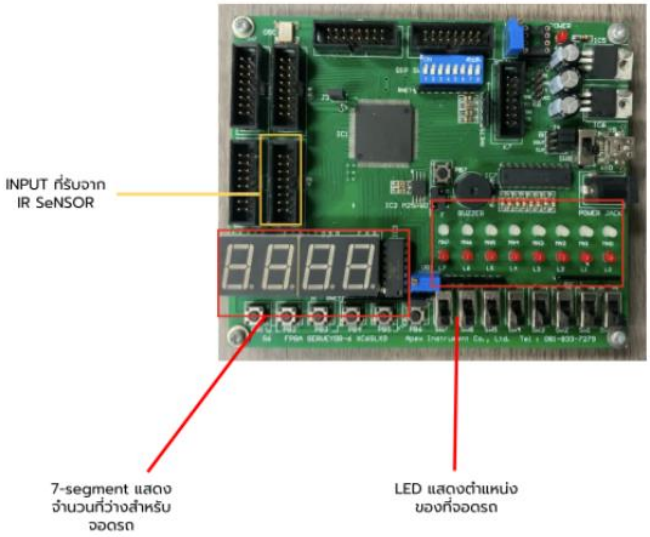
กระบวนการออกแบบ

ขั้นตอนที่ 1 การร่างวงจรที่ต้องการ

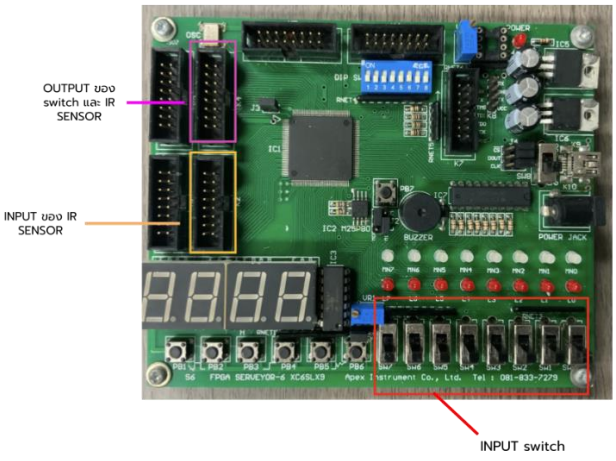
ภาพการเชื่อมต่อวงจรโดยรวม



FPGA บอร์ดที่ 1



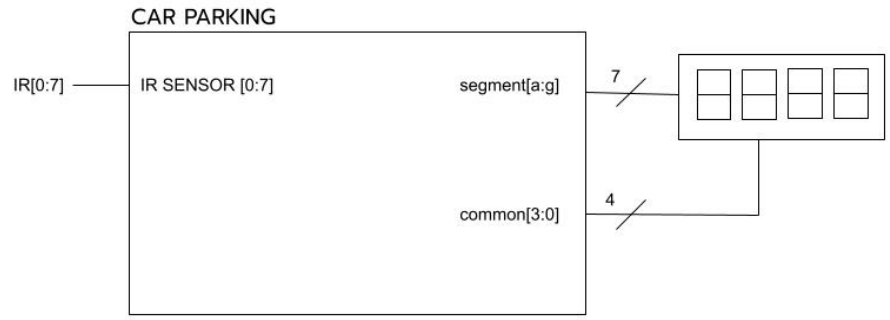
FPGA บอร์ดที่ 2



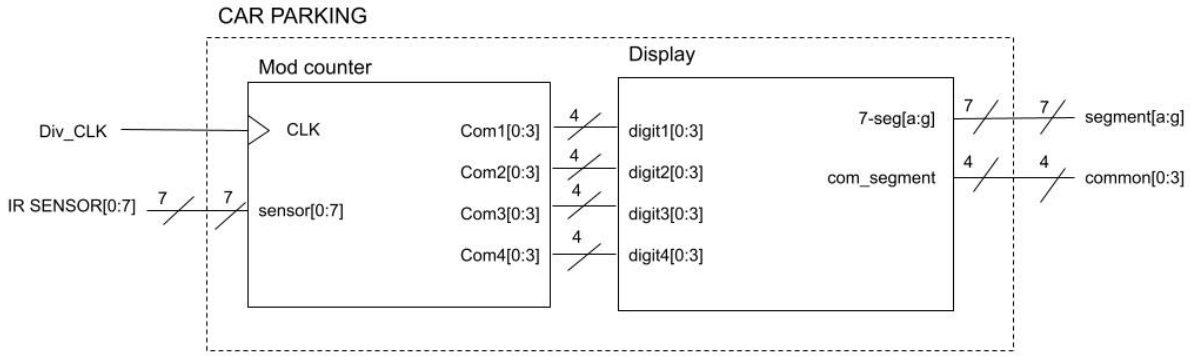
ขั้นตอนที่ 2 วาดวงจรในรูปแบบ Top down

FPGA บอร์ดที่ 1

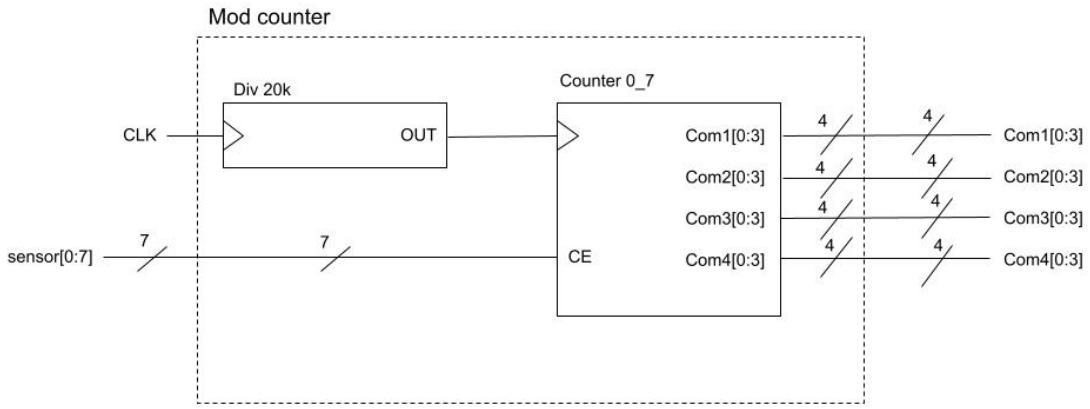
TOP LAYER



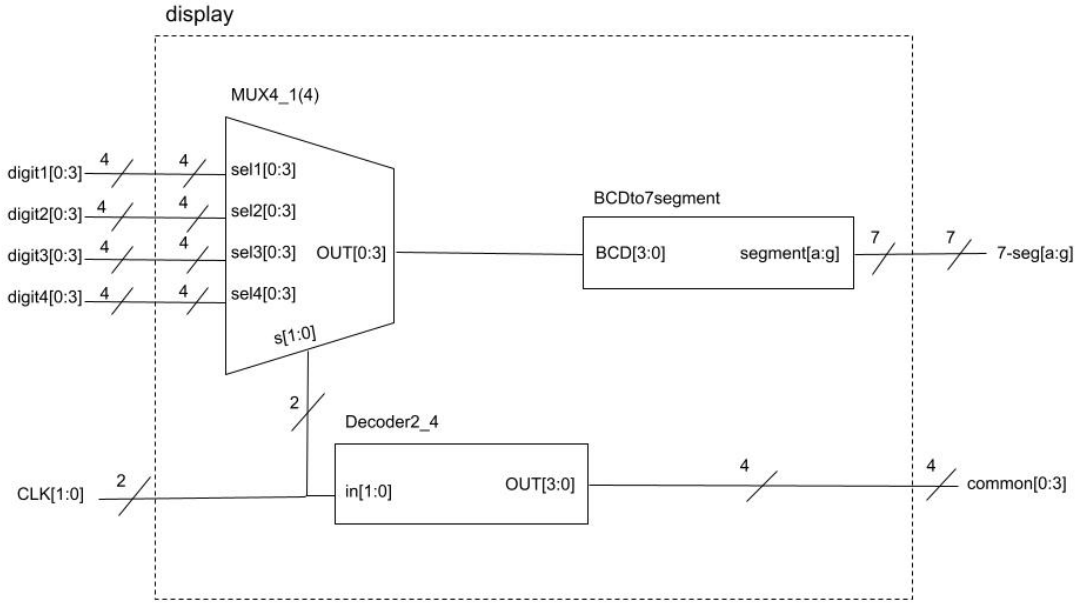
2nd LAYER



3nd LAYER(Mod counter)

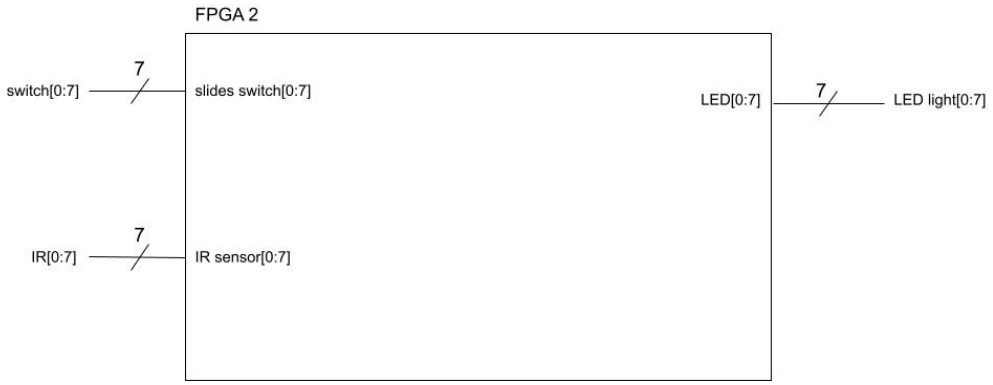


3nd LAYER(display)

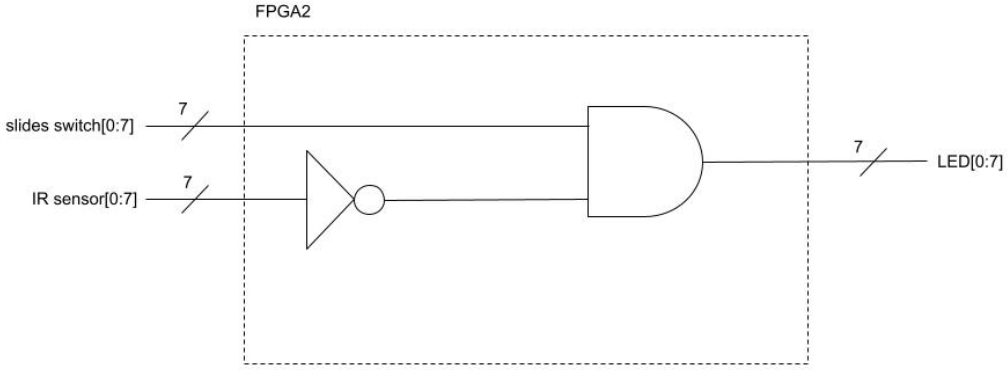


FPGA บอร์ดที่ 2

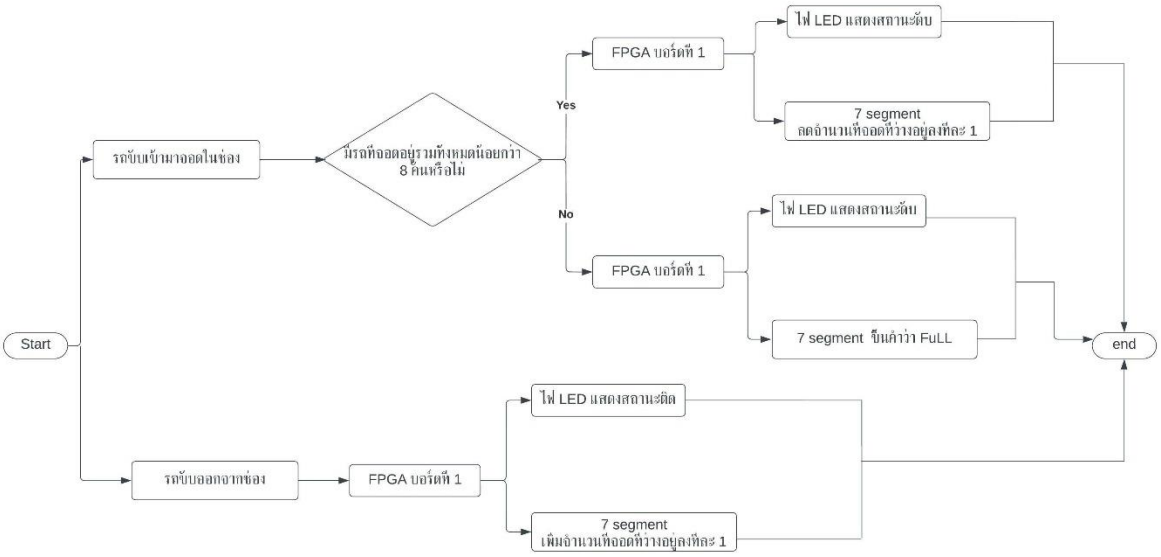
TOP LAYER



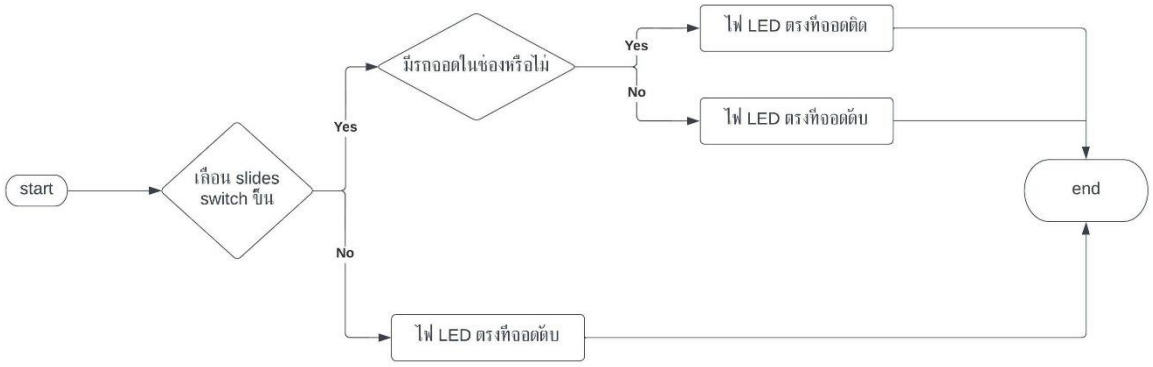
2nd LAYER



ขั้นตอนที่ 3 เขียนคำสั่งการประมวลผล FPGA บอร์ดที่ 1 (แสดงจำนวนที่ว่างของที่จอดรถและไฟ LED แสดงสถานะที่จอดรถแต่ละช่อง)



ขั้นตอนที่ 4 ต่อวงจรคำสั่งการประมวลผลบอร์ดที่ 2 (เมื่อเลื่อน slides switch ขึ้นพร้อมกับมีรถเข้ามาจอด จะทำให้ไฟ LED ในช่องจุดจอดรถนั้นๆติด)



ขั้นตอนที่ 5 เขียนคำสั่ง Arduino Uno R3 เพื่อควบคุมการเปิดปิดของคานไม้กั้นประตูทางเข้าและออก

โดยที่หากมีรถเข้าจอดเต็ม 8 คันแล้ว คานไม้กั้นประตูฝั่งทางเข้าจะไม่เปิดออกเมื่อมีรถมาจอดหยุดอยู่ตรงหน้าคานไม้กั้นประตู และจะส่งเสียงเตือนเพื่อให้ผู้ขับขี่ได้รู้

กระบวนการพัฒนา

ขั้นตอนที่ 1 กำหนดขอบเขตและระยะเวลาของการทำงาน

ลำดับที่	แผนการดำเนินโครงงาน	ระยะเวลาในการดำเนินโครงงาน							ผู้รับผิดชอบ
		พฤศจิกายน 2565				ธันวาคม 2565			
		1	2	3	4	1	2	3	
1.	ปรึกษาและเลือกหัวข้อในการทำโครงงาน	✓	✓						สมาชิกกลุ่ม
2.	ทำ Proposal โครงงาน		✓	✓	✓				สมาชิกกลุ่ม
3.	ทำ Brochure					✓	✓		สมาชิกกลุ่ม
4.	ทำ Design Document						✓		สมาชิกกลุ่ม
5.	ทำชิ้นงาน						✓		สมาชิกกลุ่ม
6.	อัดวีดีทัศน์แนะนำชิ้นงาน						✓		สมาชิกกลุ่ม

ขั้นตอนที่ 2 ลงมือปฏิบัติงานตามขั้นตอนที่ได้วางไว้

<u>บอร์ด FPGA บอร์ดที่ 1</u> (แสดงจำนวนที่ว่างของที่จอดรถและไฟ LED แสดงสถานะที่จอดรถแต่ละช่อง)

วิธีการทำงาน : เมื่อมีรถเข้ามาจอดตามช่องแต่ละช่องจะทำให้จำนวนที่ว่างที่แสดงผลผ่าน 7 Segment ลดลงและไฟแสดงสถานะของช่องนั้นๆดับ

เงื่อนไข : หากมีรถจะทำให้ไฟ LED บนบอร์ดดับและจำนวนที่แสดงผลบน 7 Segment ลดลงตามจำนวนของรถที่เข้ามาจอด หากรถที่จอดอยู่ออกไปจะทำให้ไฟ LED บนบอร์ดกลับมาติดและจำนวนที่แสดงผลบน 7 Segment เพิ่มตามจำนวนของรถที่ออกไป

แนวคิด : ส่วนการแสดงผลจำนวนที่จอดรถที่ว่างอยู่ผ่าน 7 segment จะทำการเขียน case ดักไว้ตามจำนวนของ IR sensor โดยมีทั้งหมด 8 กรณีด้วยกัน เช่น

กรณีที่ 1 มีรถเข้ามาจอด 1 คัน ก็จะให้แสดงผลเป็นเลข 7 ผ่าน 7 segment

กรณีที่ 2 มีรถเข้ามาจอด 2 คัน ก็จะให้แสดงผลเป็นเลข 6 ผ่าน 7 segment

กรณีที่ 3 มีรถเข้ามาจอด 3 คัน ก็จะให้แสดงผลเป็นเลข 5 ผ่าน 7 segment

กรณีที่ 4 มีรถเข้ามาจอด 4 คัน ก็จะให้แสดงผลเป็นเลข 4 ผ่าน 7 segment

กรณีที่ 5 มีรถเข้ามาจอด 5 คัน ก็จะให้แสดงผลเป็นเลข 3 ผ่าน 7 segment

กรณีที่ 6 มีรถเข้ามาจอด 6 คัน ก็จะให้แสดงผลเป็นเลข 2 ผ่าน 7 segment

กรณีที่ 7 มีรถเข้ามาจอด 7 คัน ก็จะให้แสดงผลเป็นเลข 1 ผ่าน 7 segment

กรณีที่ 8 มีรถเข้ามาจอด 8 คัน ก็จะให้แสดงผลเป็นคำว่า FuLL ผ่าน 7 segment

Code ที่ใช้ในการควบคุม

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Pro is
    Port (
        IR : in std_logic_vector(0 to 7);
        OSC : in std_logic;

        LED : out std_logic_vector(0 to 7);
        SEG : out std_logic_vector(0 to 6);
        COM : out std_logic_vector(3 downto 0)
    );
end Pro;

architecture Behavioral of Pro is
    function commons_decode(num : natural range 0 to 3) return std_logic_vector is
        variable temp : std_logic_vector(3 downto 0) := "1111";
    begin
        temp(num) := '0';
        return temp;
    end function;

    type seg_array is array (3 downto 0) of std_logic_vector(0 to 6);
    type seg_decode_array is array (0 to 8) of seg_array;

    constant all_seg : seg_decode_array := (
        0 => (
            3 => "1000111",
            2 => "0011100",
            1 => "0001110",
```

```
            0 => "0001110"
        ),
        1 => (0 => "0110000", others => "0000001"),
        2 => (0 => "1101101", others => "0000001"),
        3 => (0 => "1111001", others => "0000001"),
        4 => (0 => "0110011", others => "0000001"),
        5 => (0 => "1011011", others => "0000001"),
        6 => (0 => "1011111", others => "0000001"),
        7 => (0 => "1110000", others => "0000001"),
        8 => (0 => "1111111", others => "0000001")
    );

    signal ir_sig : std_logic_vector(0 to 7) := (others => '0');
    signal sel : natural range 0 to 3 := 0;
    signal count : natural := 0;
begin
    ir_sig <= IR;
    LED <= ir_sig;
    COM <= commons_decode(sel);
    SEG <= all_seg(count)(sel);

    process (OSC)
        constant DIV_CLK : natural := 20000;
        variable mod_counter : natural range 0 to DIV_CLK := 0;
        variable count_var : natural range 0 to 8 := 0;
    begin
        if rising_edge(OSC) then
            mod_counter := mod_counter + 1;
            if mod_counter = DIV_CLK then
                mod_counter := 0;
                sel <= (sel + 1) mod 4;
            end if;
```

```
            count_var := 0;
            for i in 0 to 7 loop
                if ir_sig(i) = '1' then
                    count_var := count_var + 1;
                end if;
            end loop;

            count <= count_var;
        end if;
    end process;

end Behavioral;
```

<u>บอร์ด FPGA บอร์ดที่ 2</u> (เมื่อเลื่อน slides switch ขึ้นพร้อมกับมีรถเข้ามาจอดจะทำให้ไฟ LED ในช่องจุดจอดรถนั้นๆติด)

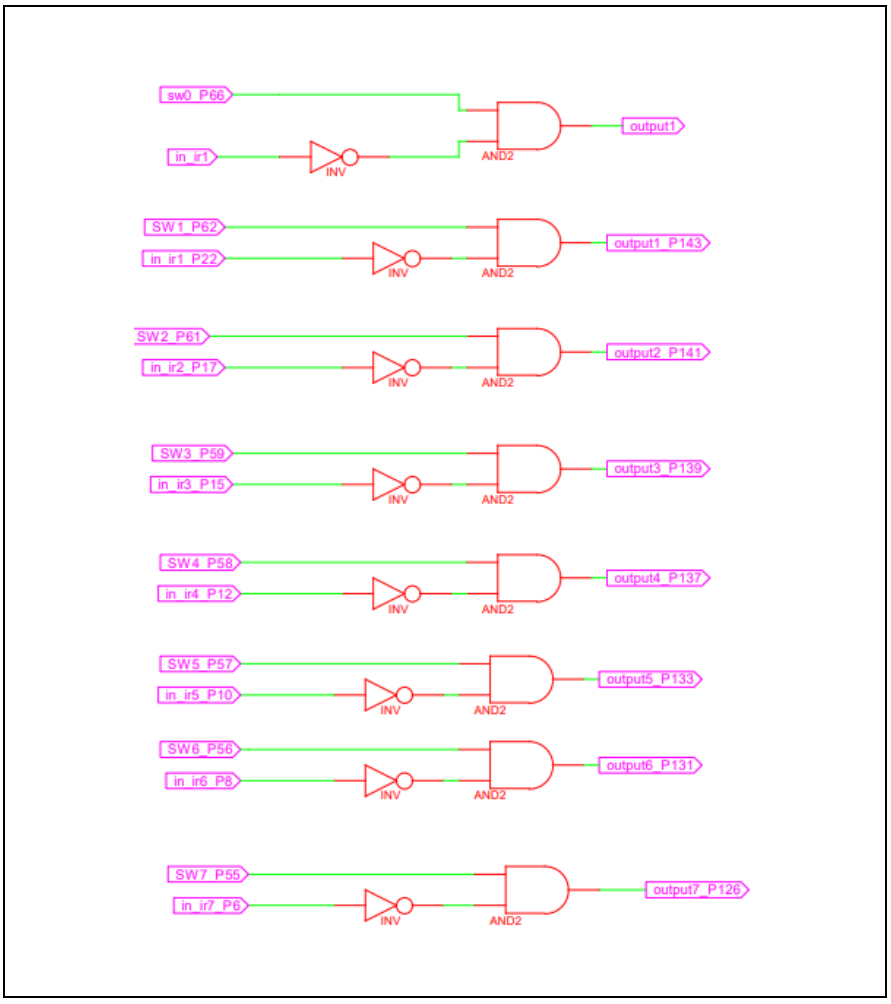
วิธีการทำงาน : เมื่อมีรถเข้ามาจอดในช่องและเลื่อน slides switch ขึ้นจะเป็นการเปิดไฟตามจุดตำแหน่งหมายเลขนั้นๆ เช่น เมื่อมีรถเข้ามาจอดในช่องที่ 2 และทำการเลื่อน slides switch 2 จะเป็นการเปิดไฟ LED ในตำแหน่งที่ 2 ดังรูป

เงื่อนไข : ไฟจะติดก็ต่อเมื่อมีรถเข้ามาจอดและเลื่อน slides switch ขึ้น

แนวคิด : รับ input เข้ามา 2 ค่า ค่าแรกจะเป็นของ slides switch แต่ละตัว ซึ่งเมื่อเลื่อนขึ้นจะเป็น 1 เลื่อนลงจะเป็น 0 ค่าที่สองที่รับมาจะเป็นของ IR sensor แต่ละตัว ซึ่งเมื่อมีรถเข้ามาจอดจะส่งค่าเป็น 0 หากไม่มีรถเข้ามาจอดจะเป็น 1 จึงทำการต่ออินเวิร์สก่อนเข้าขา and gate เพื่อทำการสลับค่ากัน เมื่อเข้าขา and gate แล้วจะมี Output 4 กรณีดังต่อไปนี้

slides switch	IR sensor (ค่าที่อินเวิร์สแล้ว)	Output ที่ออกมา (ไฟติด/ดับ)
เลื่อนลง (0)	ไม่มีรถเข้ามาจอด (0)	ไฟดับ (0)
เลื่อนลง (0)	มีรถเข้ามาจอด (1)	ไฟดับ (0)
เลื่อนขึ้น (1)	ไม่มีรถเข้ามาจอด (0)	ไฟดับ (0)
เลื่อนขึ้น (1)	มีรถเข้ามาจอด (1)	ไฟติด (1)

ภาพวงจรการสั่งงานบนบอร์ดที่ 2



บอร์ด Arduino Uno R3 (ควบคุมการเปิดปิดของคานไม้กั้นประตูเข้า - ออก)

วิธีการทำงาน : เมื่อมีรถเข้ามาหยุดอยู่ตรงหน้าคานไม้กั้นประตูเข้า-ออก (ตรง IR sensor) จะทำให้คานไม้กั้นประตูเข้า - ออกยกขึ้น

เงื่อนไข : จะต้องมีรถเข้าทางประตูทางเข้าอย่างน้อยหนึ่งคัน ประตูทางออกจึงจะสามารถทำงานได้ และหากที่จอดรถมีรถจอดครบ 8 คันแล้วมีรถคันใหม่ที่ต้องการจะเข้ามามาหยุดอยู่ตรงหน้าคานไม้กั้นประตูฝั่งทางเข้า คานไม้กั้นประตูฝั่งทางเข้าจะไม่เปิดออกและจะส่งเสียงเตือนเพื่อให้ผู้ขับขี่ได้รู้

แนวคิด : สร้างตัวแปร count ขึ้นมาเพื่อเก็บค่า โดยเมื่อมีการใช้ IR sensor ตัวที่ 1 (ฝั่งประตูทางเข้า) จะทำการนับ count เพิ่มทีละ 1 และเมื่อมีการใช้ IR sensor ตัวที่ 2 (ฝั่งประตูทางออก) จะทำการลดค่า count ลงทีละ 1

Code ที่ใช้ในการควบคุม

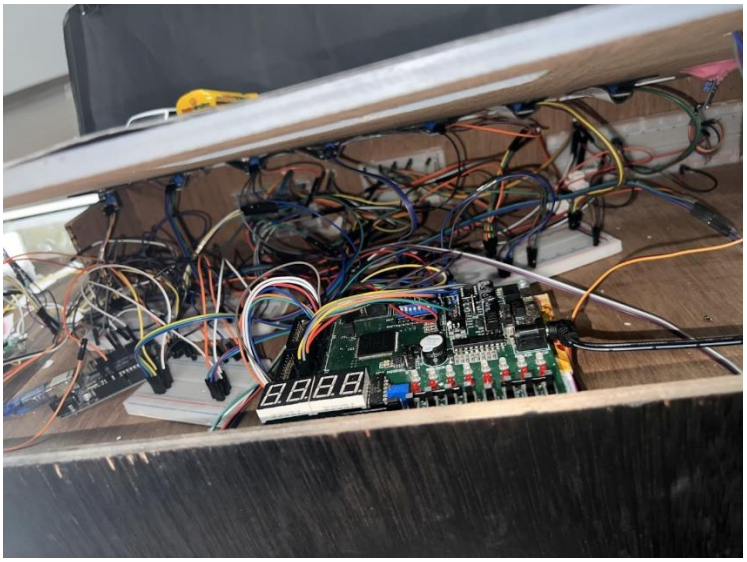
```
#include <LiquidCrystal.h>// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);
#include <Servo.h> //includes the servo library

#define  C     2100
#define  D     1870
#define  E     1670
#define  f     1580
#define  G     1400
#define  R     0

int speakerOut = 11;
int DEBUG = 1;

Servo myservo1;
#define servo 8

Servo myservo2;
#define servo2 9

int ir_s1 = 3;
int ir_s2 = 4;

int count = 0;
int count1 = 0;
int melody[] = {E };

int MAX_COUNT = sizeof(melody) / 2;
```

```
long tempo = 9000;
int pause = 600;
int rest_count = 100;
int tone_ = 0;
int beat = 0;
long duration  = 0;

void setup()
{
  pinMode(ir_s1, INPUT);
  pinMode(ir_s2, INPUT);

  pinMode(speakerOut, OUTPUT);
  if (DEBUG)
  {
    Serial.begin(9600);
  }

  myservo1.attach(8);
  myservo2.attach(9);

  myservo1.write(0);
  myservo2.write(0);
}

void loop()
{
  if (digitalRead (ir_s1) == LOW)
  {
    if (count < 8)
    {
      delay(1000);
```

```
      myservo1.write(120);
      delay(2000);
      myservo1.write(0);
      count += 1;
      Serial.print("jod ");
      Serial.print(count );
    }
    else if (count >= 8)
    {
      myservo1.write(0);
      for (int i = 0; i < MAX_COUNT; i++)
      {
        tone_ = melody[i];
        beat = 50;
        duration = beat ; // Set up timing
        playTone();
//        delayMicroseconds(pause);
        if (digitalRead (ir_s2) == HIGH)
      {
         delayMicroseconds(pause);
      }
      }
    }
  }
  if (digitalRead(ir_s2) == LOW)
  {
    if (count <= 8 && count > 0)
    {
      delay(1000);
      myservo2.write(120);
      delay(2000);
      myservo2.write(0);
```

```
      count -= 1;
      Serial.print("out ");
      Serial.print(count);
    }
    else if(count <= 0)
    {
      Serial.print("out :: ");
      Serial.print(count);
      count = 0;
    }
  }

}
void playTone() {
  long elapsed_time = 0;
  if (tone_ > 0)
  {
    while (elapsed_time < duration)
    {
      digitalWrite(speakerOut, HIGH);
      delayMicroseconds(tone_ / 3);
      digitalWrite(speakerOut, LOW);
      delayMicroseconds(tone_ / 3);
      elapsed_time += (tone_);
    }
  }
  else {
    for (int j = 0; j < rest_count; j++) {
      delayMicroseconds(duration*2);
    }
  }
}
```

ภาพการเชื่อมต่อของทั้งโครงงาน



ตัวอย่างการใช้งาน ในกรณีที่ที่จอดรถเต็มและมีรถคันใหม่ที่ต้องการจะเข้ามาจอด

เหตุผล ของการออกแบบและพัฒนาชิ้นงาน

ในปัจจุบันบางสถานที่จอดรถยังเป็นระบบจอดรถแบบธรรมดาที่ไม่มีการแสดงผลจำนวนที่ว่างที่สามารถนำรถเข้าไปจอดได้ ทำให้ในหลายๆเหตุการณ์ต้องขับรถเข้าไปวนหาที่จอดรถทำให้เป็นการใช้เวลาไปโดยเปล่าประโยชน์แถมยังเป็นการเพิ่มมลพิษในสถานที่จอดรถอีกด้วย ดังนั้นเราจึงทำการจำลองโมเดลสถานที่จอดรถขึ้นมาเพื่อเป็นแนวทางสำหรับที่จอดรถในอนาคต

ขั้นตอนที่ 3 เก็บรายละเอียดและตกแต่งชิ้นงานให้สวยงาม

ขั้นตอนที่ 4 การทดสอบผลงานและแก้ไข

ทดลองใช้งานและตรวจสอบความถูกต้องของผลงาน เพื่อให้แน่ใจว่าผลงานที่พัฒนาขึ้น ทำงานได้ถูกต้องตรงกับความต้องการที่ระบุไว้ในเป้าหมายและมีประสิทธิภาพ

ขั้นตอนที่ 5 แนวทางการพัฒนาโครงงานในอนาคตและข้อเสนอแนะ

แนวทางในการพัฒนาโครงงานในอนาคตอาจจะทำให้ไม้กั้นประตูทางเข้าออกสามารถควบคุมได้ด้วยบอร์ด FPGA ได้เลยโดยไม่ต้องใช้บอร์ด Arduino Uno R3 และเพิ่มให้มีระบบคิดเงินตามตามอัตราเวลาที่รถแต่ละคันเข้าจอด พร้อมกับแสดงผลระยะเวลาที่รถแต่ละคันจอดให้ผู้ใช้ได้ทราบ

กระบวนการทดสอบ

1. เริ่มจากการเช็คว่าวงจรทำงานได้หรือไม่

โดยส่วนแรกคือ ตัววงจรของที่จอด ทำการเช็คว่า sensor ทำงานครบหรือไม่ โดยการเอารถเข้าไปจอดทีละคันและดูที่ 7 segment ว่าเลขลดลงไปเรื่อยๆไหม และสุดท้ายคือเอารถเข้าไปจอดคัน 7segment ต้องขึ้น FuLL และไฟในบอร์ด FPGA ที่จอดไหนที่มีรถจอดไฟจะดับ และที่ไหนที่ว่างไฟจะติด ถ้าจอดครบทุกคันไฟทั้งหมดจะดับ

ส่วนที่สองคือ วงจรของสวิตซ์ไฟ ทำการเช็คว่า sensor ทำงานหรือไม่ในแต่ละที่ โดยการลองทำการกดเปิดสวิตซ์ ถ้ามีรถจอดและไฟติดคือวงจรไม่มีปัญหา

ส่วนที่สามคือ ส่วนของ Arduino Uno R3 ไม้กั้น ทำการเช็คว่า sensor ทำงานหรือไม่ โดยการนำรถขับผ่านแล้วไม้กั้นเปิด-ปิดทั้งทางเข้าและทางออก และเมื่อเข้าเกิน 8 คัน Buzzer ต้องส่งเสียง

2. ทำการเช็คภาพรวมของทุกวงจรรวมกัน

โดย 1. นำรถเข้าไปจอด 7segment ต้องแสดงจำนวนที่ลดลง ไฟแสดงตำแหน่งที่จอดต้องดับด้วย

2. ทดสอบเปิดสวิตซ์ไฟตามจุดที่มีรถจอด

3. นำรถออก 7 segment ต้องแสดงจำนวนที่เพิ่มขึ้น และไฟแสดงตำแหน่งต้องกลับมาติด

4. นำรถเข้าให้ครบ 8 คันและจอดให้ครบทุกที่ 7 segment ต้องขึ้นคำว่า FuLL และไฟแสดงตำแหน่งของที่จอดรถต้องดับทั้งหมด

5. เปิดสวิตซ์ไฟประจำที่จอดให้ครบทุกดวง เมื่อนำรถออกแล้วไฟจะต้องดับ

6. ปิดสวิตซ์ไฟประจำที่จอดให้ครบทุกดวง เมื่อนำรถเข้าแล้วไฟจะต้องดับ

ข้อมูลโครงงาน

1. วีดีทัศน์แนะนำชิ้นงาน

    Video Smart Car Parking | Youtube

    QR Code

2. ไฟล์ควบคุม FPGA

    FPGA บอร์ดที่ 1 | Google Drive

    FPGA บอร์ดที่ 2 | Google Drive

3. ไฟล์ code ที่ใช้ควบคุม Arduino Uno R3

    Smart Car Parking | ino