Sunflower

จัดทำโดย

ธนธร แตงอ่อน 64010315 กลุ่ม 118

บดินทร์ภัทร์ ราชัย 64010451 กลุ่ม 118

อนาวิล ธรรมเจริญทิพย์ 64010965 กลุ่ม 120

กฤตพร บุริยเมธากุล 64011041 กลุ่ม 120

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา

วิชา 01076006 Digital System Fundamental

เสนอ

รศ.ดร.เจริญ วงษ์ชุ่มเย็น

ปีการศึกษา 2565

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คำนำ

เนื่องจากในปัจจุบันเกิดปัญหาด้านพลังงงานไฟฟ้าซึ่งเป็นปัญหาที่สำคัญ คณะผู้จัดทำจึงได้เลือกทำโครงงานการผลิตพลังงานไฟฟ้าให้ได้มากที่สุดจากโซล่าร์เซลล์ เนื่องจากการผลิตพลังงานไฟฟ้าจากโซล่าเซลล์ให้ได้มากที่สุดนั้นจำเป็นต้องให้รังสีของดวงอาทิตย์กระทบกับหน้าของโซล่าเซลล์ให้ได้ประมาณ 35 องศา และจากเงื่อนไขดังกล่าวทำให้การผลิตพลังงานมีเวลาจำกัดอยู่แค่ช่วงนึงของวัน เนื่องจากโลกหมุนรอบดวงอาทิตย์

จากปัญหาดังกล่าว ทำให้กลุ่มของเราจึงเลือกที่จะทำการพัฒนาระบบที่สามารถทำให้แผงโซล่าเซลล์สามารถติดตามตำแหน่งของดวงอาทิตย์ เพื่อช่วยเพิ่มการผลิตพลังงานไฟฟ้าของแผงโซล่าเซลล์ให้มีประสิทธิภาพและได้ปริมาณสูงสุด

คณะผู้จัดทำ

13 ธันวาคม พ.ศ.2565

กระบวนการหาข้อมูล

1.ขั้นตอนกำหนดหัวข้อ

ในปัจจุบันพลังงานไฟฟ้าจำเป็นต่อการใช้ชีวิตประจำวันเป็นอย่างมาก ซึ่งแหล่งการผลิตพลังงานไฟฟ้าที่เป็นที่นิยมในปัจจุบันอย่างหนึ่งก็คือการผลิตพลังงานไฟฟ้าจากโซล่าเซลล์ ซึ่งการผลิตพลังงานไฟฟ้าจากโซล่าเซลล์ให้ได้มากที่สุดนั้นจำเป็นต้องให้รังสีของดวงอาทิตย์กระทบกับหน้าของโซล่าเซลล์ให้ได้ประมาณ 35 องศา คณะผู้จัดทำจึงได้ตัดสินใจทำเครื่องหมุนโซล่าเซลล์เพื่อหมุนโซล่าเซลล์ไปหารังสีของดวงอาทิตย์ที่ทำให้โซล่าเซลล์สามารถผลิตพลังงานไฟฟ้าให้ได้มากที่สุด

2.ขั้นตอนค้นหาข้อมูล

ค้นหาการทำงานของบอร์ดFGPAเพิ่มเติมเกี่ยวกับการค้นหาบอร์ดอีกตัว ศึกษาการทำงานของเซนเซอร์รับแสง ศึกำษาการทำงานของ servo ศึกษาการใช้งานบอร์ด Arduino กับตัวบอร์ด FPGA และ syntax ในการเขียนโค้ดกับ Arduino

3.ขั้นตอนการเลือกแหล่งข้อมูล

ค้นหาแหล่งข้อมูลที่มีความน่าเชื่อถือ เช่น เว็บ AMD Xilinx เว็บ Arduino

4.ขั้นตอนการเตรียมอุปกรณ์

เริ่มวิเคราะห์จากสิ่งที่จะทำ โดยแบ่งอุปกรณ์ออกเป็นทั้งหมด2ส่วน คือ

- อุปกรณ์อิเล็กทรอนิกส์ ได้แก่ บอร์ด FGPA 2 ตัว, บอร์ด Arduino, LDR, 7-Segment, Jumper Wire, Solar Cell, Servo 2 ตัว
- อุปกรณ์ส่วน Packaging ได้แก่ ไม้อัด, เทปกาวสองหน้า, น็อต, กล่องพลาสติกใส่ชิ้นงาน, 3D Printที่หมุนโซล่าเซลล์

5.ขั้นตอนการเก็บรวบรวมข้อมูล

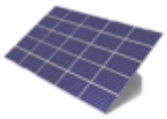
รวบรวมข้อมูลทุติยภูมิและนำมาวิเคราะห์เก็บไว้เป็นแหล่งข้อมูล

6.ขั้นตอนพิจารณาและสรุป

เป็นขั้นตอนสุดท้ายของการค้นหาข้อมูล และรวบรวมข้อมูล นำข้อมูลมาพิจารณาและหาข้อสรุปตามที่ต้องการ ตามหัวข้อที่กำหนด จากนั้นจึงนำเสนอข้อมูล

กระบวนการออกแบบ

โหมดควบคุมด้วยปุ่มกดติดปล่อยดับ (Manual)



Solar Panel

ผลิตพลังงานจากแสงอาทิตย์

Arduino Mega

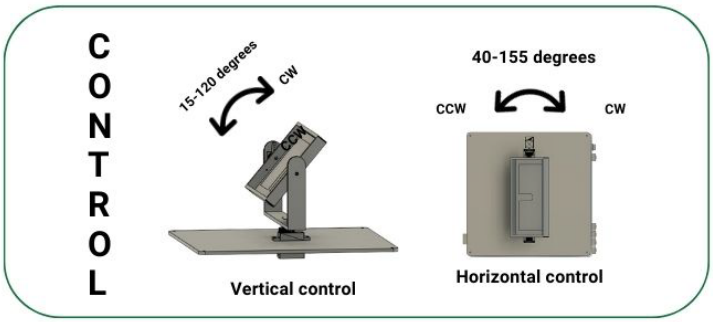
อ่านค่าจากการกดปุ่ม และอ่านค่าจาก Solar cell เพื่อทำการประมวลผลเพื่อหมุน Solar panel ไปตามปุ่มที่ถูกกด และส่งค่านั้นไปให้ Motor FPGA หมุนให้ แล้วยังมีการส่งค่าพลังงานที่ผลิตได้ไปยัง Monitor FPGA

Control Button

Increase Button : เพิ่มองศาการหมุนของเซอร์โว

Decrease Button : ลดองศาการหมุนของเซอร์โว

Change Mode Button : สลับโหมดระหว่าง Auto -> Horizontal -> Vertical



Motor FPGA

รับค่าจาก Arduino mega และทำการควบคุมเซอร์โวมอเตอร์

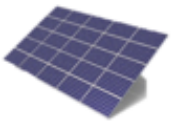
Servo Motor

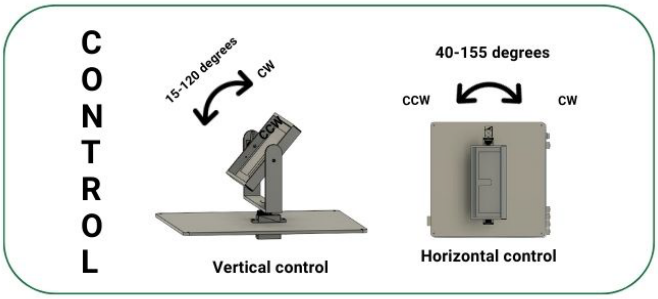
รับค่าจาก Motor FPGA และทำการหมุน Solar Panel ไปในทิศทางนั้น

Monitor FPGA

รับค่าการผลิตพลังงานจาก Arduino Mega และนำค่านั้นไปแสดงผลผ่าน 7segment

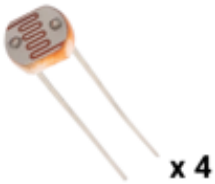
โหมดควบคุมอัตโนมัติ (Auto)





Solar Panel

ผลิตพลังงานจากแสงอาทิตย์



LDR Sensor

อ่านค่าจากแสงอาทิตย์และส่งให้ Arduino Mega

Arduino Mega

อ่านค่าจากการกดปุ่ม และอ่านค่าจาก LDR Sensor เพื่อทำการประมวลผลเพื่อหมุน Solar panel ไปตามปุ่มที่ถูกกด และส่งค่านั้นไปให้ Motor FPGA หมุนให้ แล้วยังมีการส่งค่าพลังงานที่ผลิตได้ไปยัง Monitor FPGA

Motor FPGA

รับค่าจาก Arduino mega และทำการควบคุมเซอร์โวมอเตอร์

Servo Motor

รับค่าจาก Motor FPGA และทำการหมุน Solar Panel ไปในทิศทางนั้น

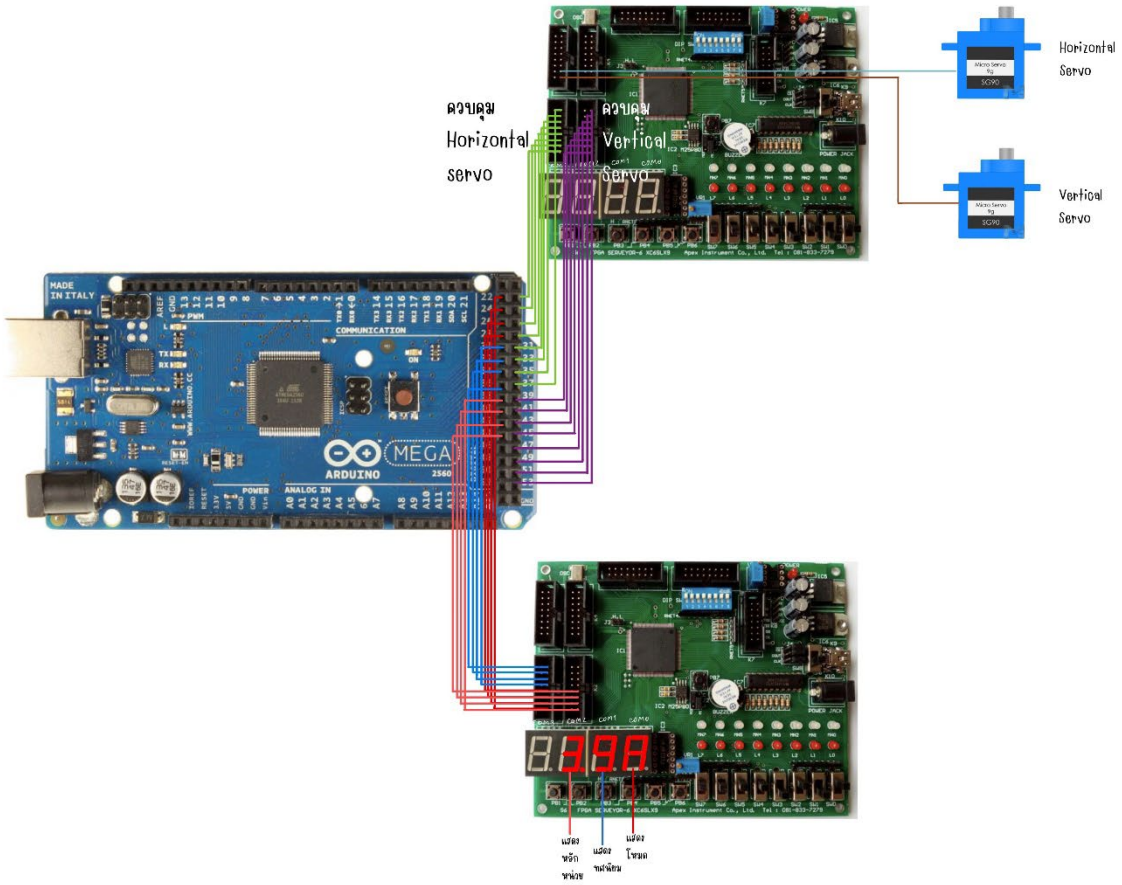
Monitor FPGA

รับค่าการผลิตพลังงานจาก Arduino Mega และนำค่านั้นไปแสดงผลผ่าน 7segment

7segment

แสดงค่าการผลิตพลังงานที่ FPGA ส่งให้เป็นหน่วยโวลต์ และโหมดปัจจุบัน

Circuit Diagram



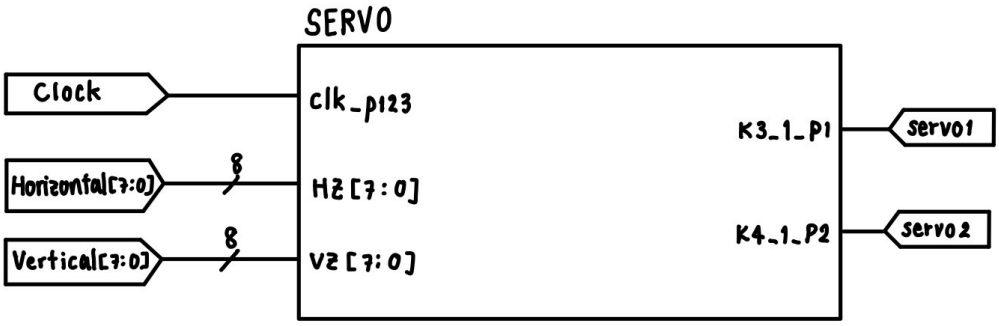
ตัวอย่าง Code Arduino

```
// LDR pin connections
int ldrTR = A0;  // LDR top right
int ldrTL = A1;  // LDR top left
int ldrBR = A2;  // LDR bottom right
int ldrBL = A3;  // LDR bottom left

// Servo horizontal;
int servoh = 90;

int servohLimitHigh = 155;
int servohLimitLow = 40;

// Servo vertical;
int servov = 90;

int servovLimitHigh = 120;
int servovLimitLow = 15;

int mode = 0;
bool isPressed = false;

//            -------- Vertical  ----------- --------- Horizontal ------------  --------------  --------------
//            LSB                          MSB LSB                          MSB LSB          MSB LSB          MSB
int pin[24] = { 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 22, 24, 26, 28, 30, 32, 34, 36 };

void servo(int vertical, int horizontal) {

void segment(int unit, int floating) {

void modeCheck(int mode) {


void automaticSolarTracker() {

void controlHorizontal() {

void controlVertical() {

void calculateVoltage() {

void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 16; i++)
    pinMode(pin[i], OUTPUT);

  pinMode(ldrTR, INPUT);
  pinMode(ldrTL, INPUT);
  pinMode(ldrBR, INPUT);
  pinMode(ldrBL, INPUT);
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
}

void loop() {

  int stateMode = digitalRead(3);
  if (stateMode == 0 && !isPressed) {
    isPressed = true;
    mode = mode + 1;
    if (mode > 2) {
      mode = 0;
    }
  }
  else if (stateMode == 1) {
  if (mode == 0) {
    modeCheck(mode);
    automaticSolarTracker();
  }
  if (mode == 1) {
    modeCheck(mode);
    controlHorizontal();
  }
  if (mode == 2) {
    modeCheck(mode);
    controlVertical();
  }
  if (mode == 3) {
    servo(0, 0);
  }
  calculateVoltage();
}
```

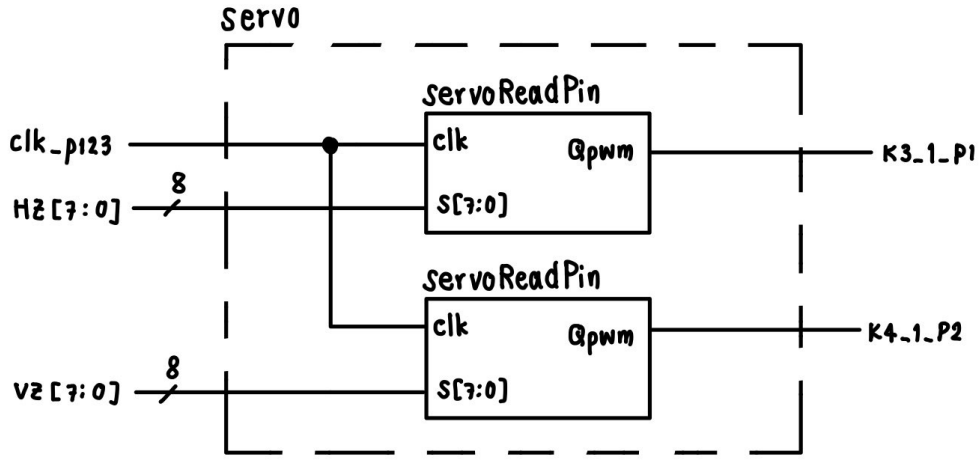
arduino code

ขั้นตอนการออกแบบ Top Down Design ของบอร์ด FPGAตัวที่หนึ่ง

First Layer



Second Layer



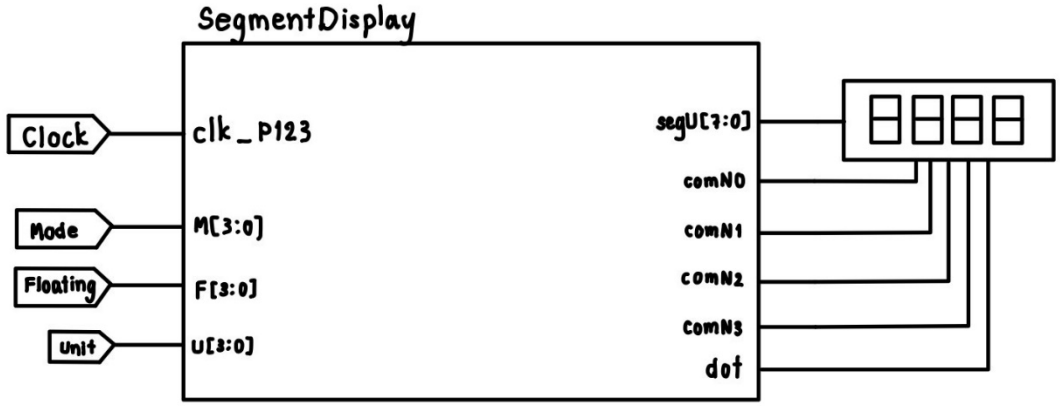
Third Layer ของ servoReadPin

ตัวอย่าง Code VHDL servoReadPin

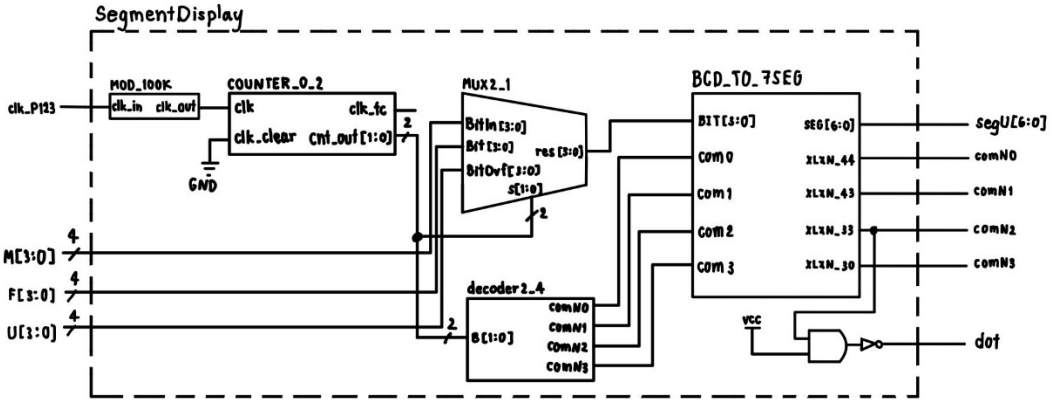
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity servoReadPin is
    Port (      clk : in  STD_LOGIC;

                                s : in STD_LOGIC_VECTOR(7 DOWNTO 0);

                                Qpwm : out  STD_LOGIC

                        );

end servoReadPin;

architecture Behavioral of servoReadPin is

   signal COUNT : integer range 0 to 2048 ;

        signal sq : std_logic;

        signal spwm :  STD_LOGIC_VECTOR(174 DOWNTO 0);

        component DIVIDER is

                port
                (
                                CLK : in std_logic;

                                Q : out std_logic
                );

   end component;

begin

process (sq)

begin

               if sq'event and sq = '1' then

                       if (COUNT >= 2000) then

                               COUNT <= 0;

                       else

                               COUNT <= COUNT +1;

                       end if;

               end if;

end process;

process (COUNT)

        begin

                if (COUNT <= 100) then spwm(0) <= '1'; else spwm(0) <= '0'; end if;
                if (COUNT <= 101) then spwm(1) <= '1'; else spwm(1) <= '0'; end if;
                if (COUNT <= 102) then spwm(2) <= '1'; else spwm(2) <= '0'; end if;
```

ขั้นตอนการออกแบบ Top Down Design ของบอร์ด FPGAตัวที่สอง

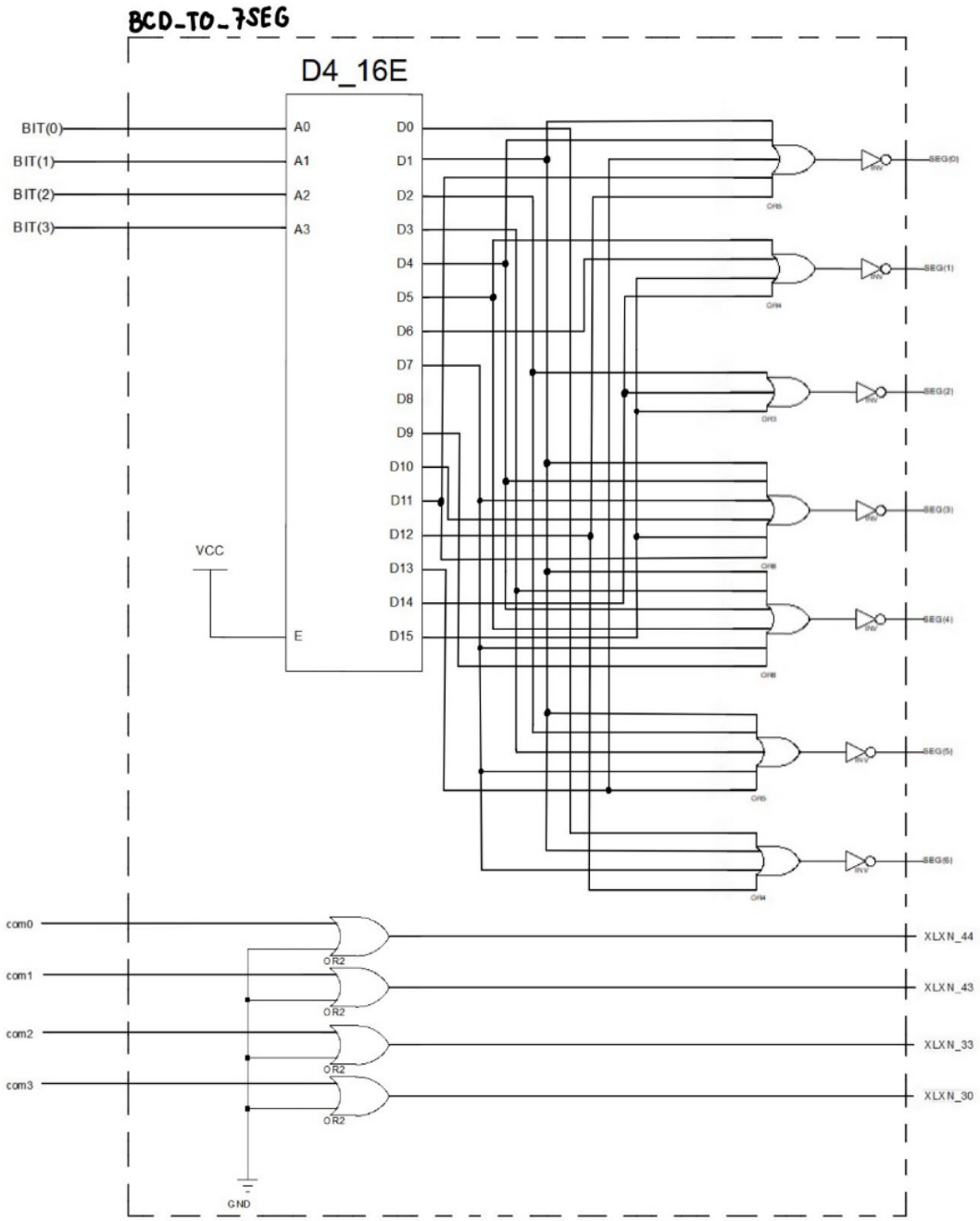
First Layer



Second Layer



Third Layer ของ BCD_TO_7SEG



Third Layer ของ MOD_100K

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

entity MOD_100K is
   port (
      clk_in : in std_logic;
      clk_out : out std_logic
   );
end MOD_100K;

architecture Behavioral of MOD_100K is
begin
   process (clk_in)
      variable counter : natural;
      variable c_out : std_logic := '0';
   begin
      if rising_edge(clk_in) then
         counter := counter + 1;
         if counter = 50000 then
            counter := 0;
            c_out := NOT(c_out);
         end if;
      end if;
      clk_out <= c_out;
   end process;
end Behavioral;
```

Third Layer ของ COUNTER0_2

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

entity COUNTER_0_2 is
   port (
      clk : in std_logic;
      cnt_out : out std_logic_vector (1 downto 0);
      clk_tc : out std_logic;
      clk_clear : in std_logic
   );
end COUNTER_0_2;

architecture Behavioral of COUNTER_0_2 is
begin
   process (clk,clk_clear)
      constant COUNT_INCREMENT : integer := 1;
      constant COUNT_START : integer := 0;
      constant COUNT_END : integer := 2;
      constant COUNT_TERMINAL : natural := 2;
      variable counter : integer := COUNT_START;
   begin
      cnt_out <= std_logic_vector(to_unsigned(counter, cnt_out'length));
      if clk_clear = '1' then
            counter := COUNT_START;
      elsif rising_edge(clk) then
         counter := counter + COUNT_INCREMENT;
         if counter = COUNT_END + COUNT_INCREMENT then
            counter := COUNT_START;
         end if;
         if counter = COUNT_TERMINAL then
            clk_tc <= '1';
         else
            clk_tc <= '0';
         end if;
      end if;
   end process;
end Behavioral;
```

กระบวนการพัฒนา

การเตรียมการ

เตรียมเครื่องคอมพิวเตอร์ ซอฟต์แวร์และวัสดุอุปกรณ์ต่างๆที่จะใช้ในการทำโครงงาน เพิ่มเติมเพื่อใช้ในการพัฒนาชิ้นงาน

การลงมือพัฒนา

อัพโหลดโค้ดจาก Arduino ส่งให้ตัวบอร์ด FPGA และ อัพโหลดรันโค้ดทั้งหมด เพื่อให้ FPGA ทำการสั่ง servo พบปัญหาในการสั่ง servo ที่องศาผิดเพี้ยนไป แก้ไขโดยการmapค่าองศาที่จะใช้ในการสั่งservo เพื่อให้ค่าที่ทำการหมุนไม่ผิดเพี้ยนไป จนแก้ไขได้ในที่สุด

การทดสอบผลงานและแก้ไข

ทำการรันโค้ดทั้งหมดเพื่อตรวจสอบความถูกต้อง เพื่อเทสวงจร และ หากเกิดข้อผิดพลาดในการทำงานก็ทำการตรวจหาสาเหตุ

การอภิปรายและข้อเสนอแนะ

ใช้บอร์ด Arduino ในการประมวลผลและส่งข้อมูลการทำงานให้กับบอร์ด FPGAตัวแรก ในการควบคุมการทำงานของ servo และ FPGAตัวที่สองในส่วนของการแสดงผลค่าพลังงานไฟฟ้าในหน่วยโวลต์และโหมดการทำงานต่างๆ

การเตรียมการ

เตรียมเครื่องคอมพิวเตอร์ ซอฟต์แวร์และวัสดุอุปกรณ์ต่างๆที่จะใช้ในการทำโครงงาน เพิ่มเติมเพื่อใช้ในการพัฒนาชิ้นงาน

การลงมือพัฒนา

อัพโหลดโค้ดจาก Arduino ส่งให้ตัวบอร์ด FPGA และ อัพโหลดรันโค้ดทั้งหมด เพื่อให้ FPGA ทำการสั่ง servo พบปัญหาในการสั่ง servo ที่องศาผิดเพี้ยนไป แก้ไขโดยการmapค่าองศาที่จะใช้ในการสั่งservo เพื่อให้ค่าที่ทำการหมุนไม่ผิดเพี้ยนไป จนแก้ไขได้ในที่สุด

การทดสอบผลงานและแก้ไข

ทำการรันโค้ดทั้งหมดเพื่อตรวจสอบความถูกต้อง เพื่อเทสวงจร และ หากเกิดข้อผิดพลาดในการทำงานก็ทำการตรวจหาสาเหตุ

การอภิปรายและข้อเสนอแนะ

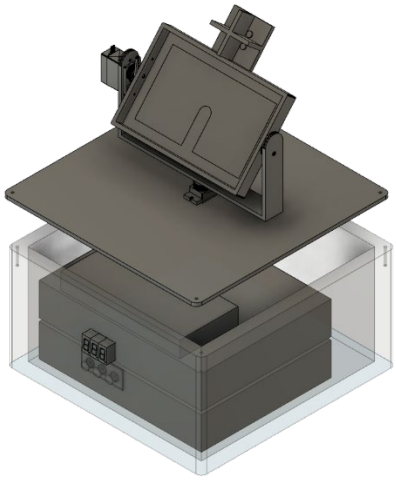
ใช้บอร์ด Arduino ในการประมวลผลและส่งข้อมูลการทำงานให้กับบอร์ด FPGAตัวแรก ในการควบคุมการทำงานของ servo และ FPGAตัวที่สองในส่วนของการแสดงผลค่าพลังงานไฟฟ้าในหน่วยโวลต์และโหมดการทำงานต่างๆ

แนวทางการพัฒนาโครงงานในอนาคตและข้อเสนอแนะ

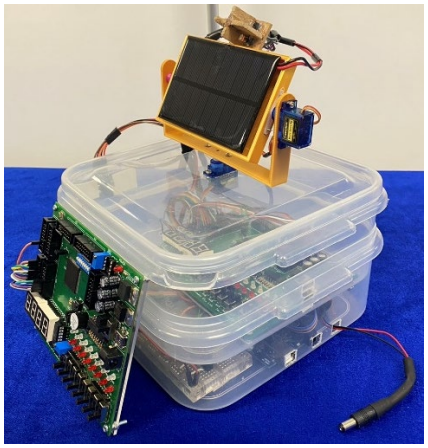
โปรแกรม Xilinx ที่ใช้ในการวาดวงจร อาจมีปัญหาทางด้านซอฟต์แวร์ ทำให้การทำงานไม่ค่อยเสถียร แต่การเขียนวงจรด้วยโค้ด จะไม่พบปัญหามากนัก อีกทั้งยังตรวจสอบการเขียนวงจรได้ง่ายกว่ามาก

Packaging

3D Modeling

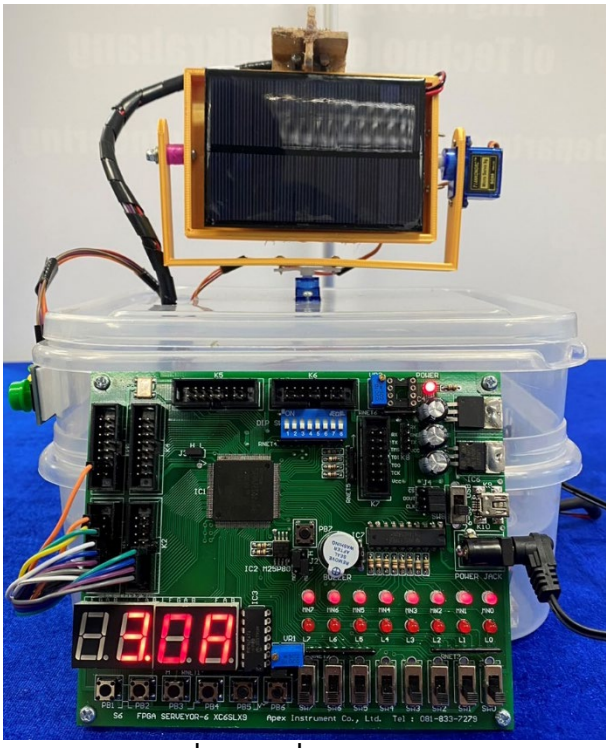


ชิ้นงานจริง

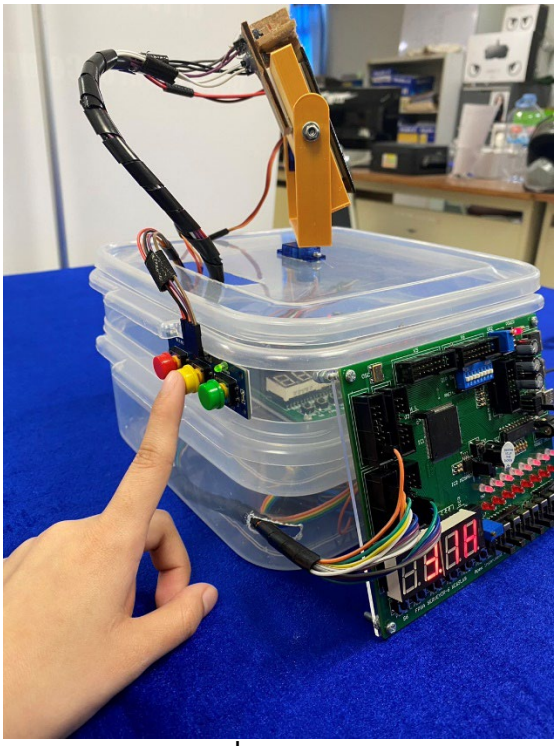


กระบวนการทดสอบ

1. เมื่อเปิดระบบจะเข้าสู่โหมดอัตโนมัติ (Auto Mode : A) เป็นโหมดแรก (Default) และสามารถกดปุ่มสีเหลืองเพื่อสลับโหมดได้ทั้งหมด 3 โหมด (Auto -> Horizontal -> Vertical) ตามลำดับ



เมื่อเปิดเครื่อง(รูปซ้าย)



ปุ่มเปลี่ยนโหมด(รูปขวา)

2. โหมดอัตโนมัติ (Auto Mode : A)

เมื่อโซล่าเซลล์ได้รับแสง ระบบจะทำการประมวลผลเพื่อหันหันโซล่าเซลล์ไปในทิศทางนั้น และจะทำการปรับองศาของหน้าโซล่าเซลล์เพื่อให้สามารถผลิตพลังงานได้สูงที่สุด

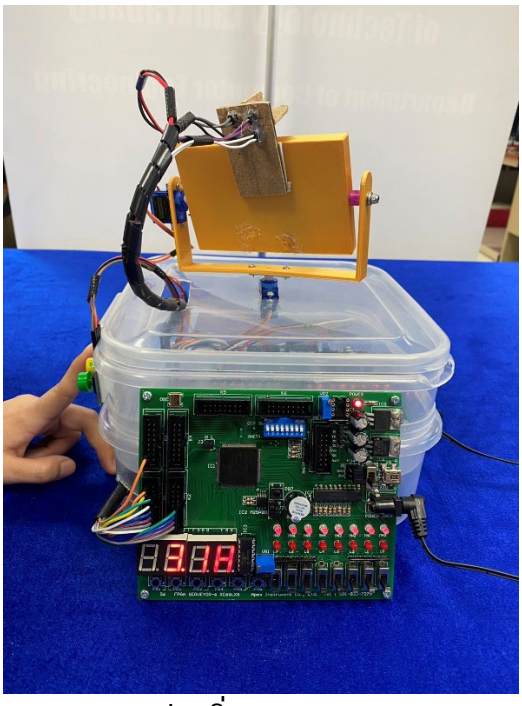
หมายเหตุ : เมื่ออยู่ในโหมดอัตโนมัติ จะไม่สามารถกดปุ่มเพิ่ม หรือลดองศาได้

โหมดอัตโนมัติ (Auto Mode : A)

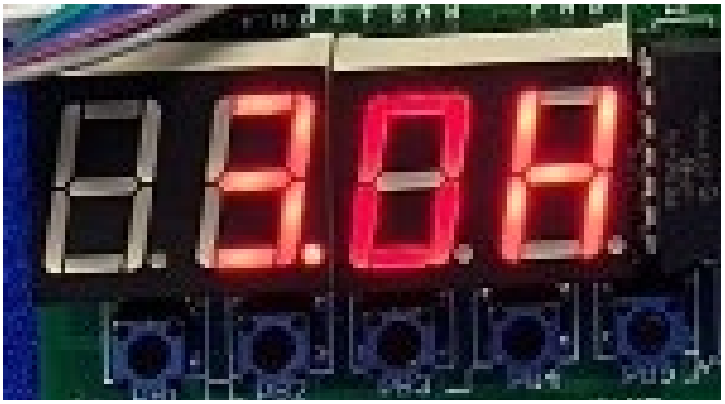
3. โหมดปรับองศาแนวราบ(Horizontal Mode : H)

ในโหมดนี้ จะสามารถใช้ปุ่มสีแดง ในการเพิ่มองศาของเซอร์โวแนวราบได้ (Horizontal Servo) และยังสามารถใช้ปุ่มสีเขียว ในการลดองศาของเซอร์โวแนวราบ ได้เช่นกัน

กดปุ่มลดองศา(รูปซ้าย)



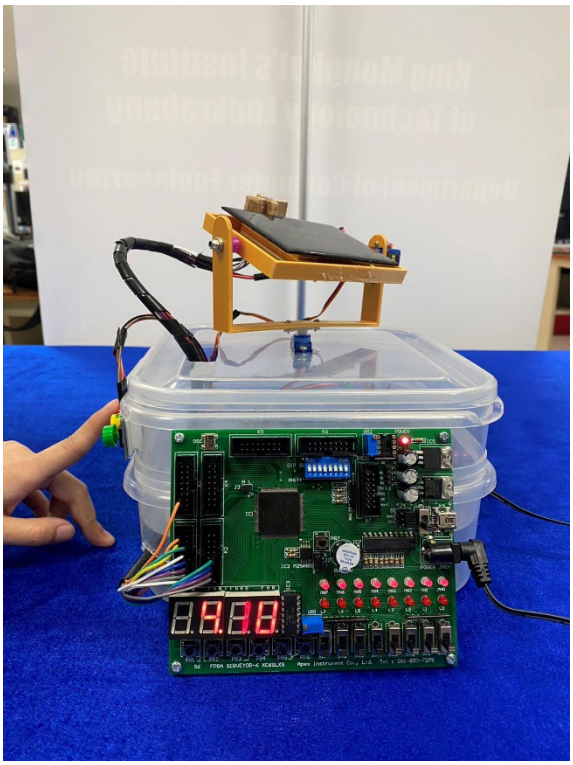
กดปุ่มเพิ่มองศา(รูปขวา)



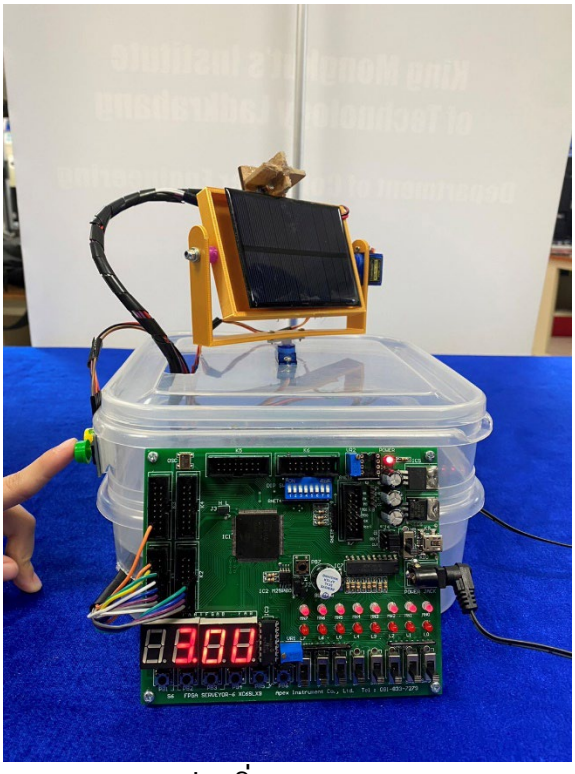
โหมดปรับองศาแนวราบ(Horizontal Mode : H)

4. โหมดปรับองศาแนวดิ่ง(Vertical Mode : U)

ในโหมดนี้ จะสามารถใช้ปุ่มสีแดง ในการเพิ่มองศาของเซอร์โวแนวดิ่งได้ (Vertical Servo) และ ยังสามารถใช้ปุ่มสีเขียว ในการลดองศาของเซอร์โวแนวดิ่ง ได้เช่นกัน



กดปุ่มลดองศา(รูปซ้าย)



กดปุ่มเพิ่มองศา(รูปขวา)

โหมดปรับองศาแนวดิ่ง(Vertical Mode : U)

5. เมื่ออยู่ในโหมดปรับองศาแนวดิ่ง(Vertical Mode : U) และกดปุ่มสีเหลืองอีกครั้ง จะกลับเป็นโหมดอัตโนมัติ (Auto Mode : A) อีกครั้ง

QR CODE วิดีทัศน์แนะนำชิ้นงานและวิธีการใช้งาน และ Google Driveรวมข้อมูลไฟล์

วิดีทัศน์แนะนำชิ้นงานและวิธีการใช้งาน

Google Drive รวมข้อมูลไฟล์