มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

รายงานวิจัยฉบับสมบูรณ์

การพัฒนาระบบควบคุมสำหรับระบบสะท้อนแสงอาทิตย์

Development of Control System for Sun Reflecting System

ดร.มนูศักดิ์ จานทอง
ผศ.ดร.บุณย์ฤทธิ์ ประสาทแก้ว
ดร.ปรัชญา เปรมปราณีรัชต์
ผศ.ณัฐสิทธิ์ พัฒนะอิ่ม

ส่วนที่ 1 รายละเอียดโครงการ

1. ชื่อโครงการวิจัย การพัฒนาระบบควบคุมสำหรับระบบสะท้อนแสงอาทิตย์
(Development of control system for sun reflecting system)

2. หน่วยงานหลักที่รับผิดชอบงานวิจัย และสถานที่ตั้งพร้อมทั้งชื่อหน่วยงาน และลักษณะของการร่วมงานวิจัยกับหน่วยงานอื่น (ถ้ามี)

หน่วยงานหลักที่รับผิดชอบ

ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี ต. คลองหก อ.ธัญบุรี จ.ปทุมธานี 12110 โทร. 0-2549-3430-5 โทรสาร 0-2549-3432

3. คณะผู้วิจัย บทบาทของนักวิจัยแต่ละคนในการทำวิจัย และสัดส่วนที่ทำการวิจัย (%)

ลำดับ	ชื่อ-สกุล	บทบาทนักวิจัย	สัดส่วนการทำวิจัย
1	ดร.มนูศักดิ์ จานทอง Dr.-Ing. Manusak Janthong	หัวหน้าโครงการและผู้วิจัยหลัก	50%
2	ผศ.ดร.บุณย์ฤทธิ์ ประสาทแก้ว Asst.Prof. Dr. Boonrit Prasartkaew	ผู้ร่วมวิจัย	10%
3	ดร.ปรัชญา เปรมปราณีรัชต์ Dr. Pradya Prempraneerach	ผู้ร่วมวิจัย	20%
4	ผศ.ณัฐสิทธิ์ พัฒนะอิ่ม Asst.Prof. Nathasit Phathanaim	ผู้ร่วมวิจัย	20%

4. ประเภทของการวิจัย การพัฒนาทดลอง

5. สาขาวิชาการและกลุ่มวิชาที่ทำการวิจัย สาขาวิศวกรรมศาสตร์และอุตสาหกรรมวิจัย

ส่วนที่ 2 เนื้อหาโครงการ

บทคัดย่อ

โครงการวิจัยนี้เป็นการพัฒนาระบบควบคุมสะท้อนแสงอาทิตย์โดยใช้ไมโครคอนโทรลเลอร์ ซึ่งใช้อัลกอลิทึม เอสพีเอ ในการคำนวณหาตำแหน่งของดวงอาทิตย์ จากนั้นแปลงเป็นมุมในการหมุนของระบบสะท้อนแสงอาทิตย์เพื่อสะท้อนแสงไปยังตำแหน่งเป้าหมายที่ต้องการ

โครงสร้างของระบบสะท้อนแสงอาทิตย์มีองศาอิสระในการหมุนเท่ากับ 2 และติดตั้งแผ่นอะครีลิคสะท้อนแสง ขนาดกว้าง 1 เมตร ยาว 1 เมตร ไว้ด้านบนของโครงสร้างเพื่อเป็นตัวสะท้อนแสงอาทิตย์ โดยใช้ดีซีเซอร์โวมอเตอร์ 24 โวลต์ เป็นตัวควบคุมการหมุนของระบบให้สะท้อนแสงไปยังเป้าหมาย ในการควบคุมการทำงานของระบบสะท้อนแสงอาทิตย์ใช้ไมโครคอนโทรลเลอร์อาร์ดูโน (Arduino) เป็นตัวประมวลผล

การทดสอบระบบสะท้อนแสงอาทิตย์ เริ่มตั้งแต่เวลา 9.00 น. ถึง 16.00 น. จากผลการทดสอบพบว่าระบบสะท้อนแสงอาทิตย์สามารถทำงานและสะท้อนแสงไปยังเป้าหมายที่ต้องการได้ตลอดช่วงเวลาที่ทดสอบ มีความคลาดเคลื่อนอยู่ที่ ±0.2 เมตร

Abstract

This research presents a development of control system for reflecting sunlight using microcontroller. The sun position calculated by SPA algorithm is determined the angle of the sun reflecting system in order to keep the sunlight on a desired target.

The structure of sun reflecting system, which has 2 DOF, is installed with an acrylic board of 1 square meter as sun reflective component. To control the rotation of the system for reflecting the sunlight to the target, 24V servo motors are used to be the actuators and Arduino microcontroller is implemented as controller.

The experiments, which start at 9.00 A.M. to 4.00 P.M., show that the system can work well and reflect the sunlight to the target with error of ±0.2 meter.

กิตติกรรมประกาศ

คณะผู้วิจัยขอขอบพระคุณมหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรีและสำนักงานวิจัยแห่งชาติ (วช.) อย่างสูงที่ได้มอบทุนวิจัยสำหรับโครงการวิจัยนี้ และขอขอบพระคุณภาควิชาวิศวกรรมเครื่องกลที่อำนวยความสะดวกพร้อมทั้งสนับสนุนในด้านต่างๆเพื่อดำเนินการโครงการวิจัย และท้ายที่สุดขอกล่าวคำขอบพระคุณไปยังคณาจารย์ประจำภาควิชาวิศวกรรมเครื่องกลทุกๆ ท่านที่ได้ให้คำแนะนำที่มีประโยชน์ทุกๆ ด้านแก่คณะผู้วิจัย

คณะผู้วิจัย

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

ปัจจุบันนี้ปัญหาเรื่องพลังงานเป็นปัญหาที่สำคัญที่ประเทศชาติกำลังเผชิญอยู่ ซึ่งประเทศไทยมีความต้องการที่จะบริโภคพลังงานในแต่ละปีเป็นจำนวนมหาศาล พลังงานส่วนใหญ่ที่ใช้กันอยู่นั้นได้มาจากฟอสซิล ได้แก่ น้ำมันดิบ ถ่านหิน และก๊าซธรรมชาติ เป็นต้น จะเห็นได้ว่าในแต่ละวันเราจะประสบกับปัญหาราคาค่าเชื้อเพลิงเพิ่มราคาสูงขึ้น ไม่ว่าจะเป็นราคาน้ำมัน ราคาก๊าซหุงต้ม ที่ใช้ในครัวเรือนกับยานพาหนะต่าง ๆและผลของการปรับขึ้นราคาเชื้อเพลิงนี้ยังมีผลกระทบไปถึงกับโรงงานอุตสาหกรรมภายในประเทศ และที่สำคัญอีกอย่างก็คือเชื้อเพลิงที่มาจากฟอสซิลเหล่านี้มีโอกาสที่จะหมดไปจากโลกนี้ จะเห็นได้ว่าปัญหาเรื่องพลังงานเป็นปัญหาสำคัญอย่างยิ่ง ซึ่งทำให้ประเทศไทยต้องเสียเงินตราไปยังต่างประเทศปีละหลายแสนล้านบาทเพื่อที่จะซื้อเชื้อเพลิงมาใช้บริโภคภายในประเทศ ดังนั้นพลังงานทางเลือกจึงเป็นพลังงานที่น่าสนใจที่ต้องศึกษาเพื่อที่จะได้นำมาช่วยทดแทนพลังงานหลักที่มาจากฟอสซิล พลังงานทางเลือกที่พบเห็นกันมากได้แก่ พลังงานจากน้ำ พลังงานจากลม พลังงานจากชีวมวลและพลังงานจากแสงอาทิตย์ ซึ่งพลังงานเหล่านี้สามารถนำมาใช้ในการผลิตกระแสไฟฟ้าทดแทนการใช้น้ำมันและแก๊ส ซึ่งจะช่วยลดการนำเข้าพลังงานจากฟอสซิล และเมื่อพิจารณาถึงปริมาณหรือศักยภาพพลังงานรังสีตรงจากแสงอาทิตย์ของประเทศไทย จะเห็นได้ว่าประเทศไทยได้รับพลังงานรังสีตรงจากแสงอาทิตย์สูงสุดอยู่ที่ 1,350-1,400 KWh/m^2-yr ซึ่งถือว่าประเทศไทยมีศักยภาพทางด้านพลังงานจากแสงอาทิตย์ค่อนข้างสูง ด้วยเหตุนี้จึงควรใช้ข้อดีนี้มาเป็นประโยชน์

และพัฒนาเทคโนโลยีที่เกี่ยวข้องกับแสงอาทิตย์ เพื่อที่นำเอาพลังงานที่ได้จากแสงอาทิตย์มาทำคุณประโยชน์ต่อประเทศชาติต่อไป

คณะผู้วิจัยจึงนำเสนอโครงงานวิจัยที่เกี่ยวข้องกับพลังงานแสงอาทิตย์ ซึ่งเป็นระบบควบคุมการเคลื่อนที่ของเสาสะท้อนแสงอาทิตย์เพื่อที่ให้สะท้อนแสงอาทิตย์ไปรวมกันที่ตำแหน่งที่ต้องการหรือไปยังตัวรับรังสีรวมกลาง (Central receiver) ซึ่งตัวรับรังสีรวมกลางนี้จะเป็นตัวรับรังสีความร้อนจากแสงอาทิตย์และถ่ายเทความร้อนนี้สู่ของไหล ที่ไหลอยู่ภายในตัวรับรังสีรวมกลาง และของไหลร้อนนี้ก็จะถูกนำไปใช้ในการผลิตกระแสไฟฟ้าต่อไป โดยที่โครงการวิจัยนี้จะทำการศึกษาวิจัยเฉพาะการออกแบบเสาสะท้อนแสงอาทิตย์ที่สามารถหมุนหรือเคลื่อนที่ได้สองแกน และออกแบบระบบควบคุมการเคลื่อนที่ของเสาสะท้อนแสงอาทิตย์ให้สะท้อนไปยังตำแหน่งที่ต้องการได้ โดยใช้ไมโครคอนโทรลเลอร์เข้ามาช่วยในการประมวลผล

วัตถุประสงค์ของโครงการวิจัย

1. เพื่อศึกษาและพัฒนาระบบควบคุมการเคลื่อนที่ของเสาสะท้อนแสงอาทิตย์ที่สามารถใช้ในขบวนการผลิตกระแสไฟฟ้า
2. เพื่อศึกษาและพัฒนาระบบกลไกการเคลื่อนที่ของเสาสะท้อนแสงอาทิตย์

ขอบเขตของโครงการวิจัย

1. สร้างเสาสะท้อนแสงอาทิตย์ที่หมุนได้ 2 แกน จำนวน 2 เสา
2. เสาสะท้อนแสงสามารถรองรับวัสดุสะท้อนแสงที่มีขนาด 1x1 เมตร
3. ออกแบบระบบควบคุมที่ควบคุมให้เสาสะท้อนแสงอาทิตย์สามารถสะท้อนแสงอาทิตย์ไปยังตำแหน่งที่ต้องการได้ โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการเคลื่อนที่

ประโยชน์ที่คาดว่าจะได้รับ

1. เป็นฐานความรู้ที่เกี่ยวข้องกับพลังงานงานแสงอาทิตย์
2. ผลการทำงานของระบบที่ออกแบบสามารถถูกนำไปใช้เป็นฐานความรู้ เพื่อที่จะช่วยในการออกแบบระบบสะท้อนรังสีดวงอาทิตย์ให้มีประสิทธิภาพมากยิ่งขึ้นต่อไปในอนาคต พร้อมกันนั้นยังช่วยพัฒนาศักยภาพเทคโนโลยีพลังงานแสงอาทิตย์ของประเทศ
3. ได้ระบบควบคุมการสะท้อนแสงอาทิตย์ที่ใช้ในขบวนการผลิตกระแสไฟฟ้า ซึ่งผู้ประกอบการผลิตไฟฟ้า อาทิเช่น การไฟฟ้าฝ่ายผลิต เป็นต้น สามารถนำไปประยุกต์หรือพัฒนาต่อยอดในการ ผลิตกระแสไฟฟ้าได้

หน่วยงานที่จะนำผลการวิจัยไปใช้ประโยชน์

1. คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
2. คณะครุศาสตร์อุตสาหกรรม มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

3. คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
4. ศูนย์พัฒนาบุคลากรเพื่ออุตสาหกรรมและปิโตรเคมี มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
5. หน่วยงานของรัฐและเอกชนที่มีความสนใจ
6. นักวิจัย นักประดิษฐ์ ของสถาบันการศึกษาต่าง ๆ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

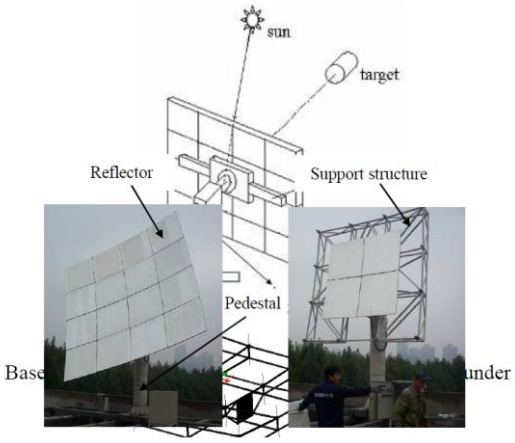
2.1 วรรณกรรมที่เกี่ยวข้อง

ในการศึกษาวิจัยงานวิจัยที่มีความเกี่ยวข้องกับการควบคุมระบบสะท้อนแสงอาทิตย์ คณะผู้วิจัยได้ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้องเพื่อใช้เป็นข้อมูลพื้นฐานในการทำวิจัย โดยรวบรวมและเรียบเรียงสาระสำคัญไว้ดังต่อไปนี้

Grena [1] ได้นำเสนออัลกอลิทึมใหม่ที่ใช้ในการหาตำแหน่งดวงอาทิตย์ ซึ่งใช้ในการติดตามดวงอาทิตย์ที่มีค่าความผิดพลาดสูงสุดที่ 0.0027 องศา ในช่วง ค.ศ. 2003-2023 ซึ่งเป็นกระบวนการหาตำแหน่งดวงอาทิตย์ที่ใช้เวลาน้อยและไม่สับซ้อน

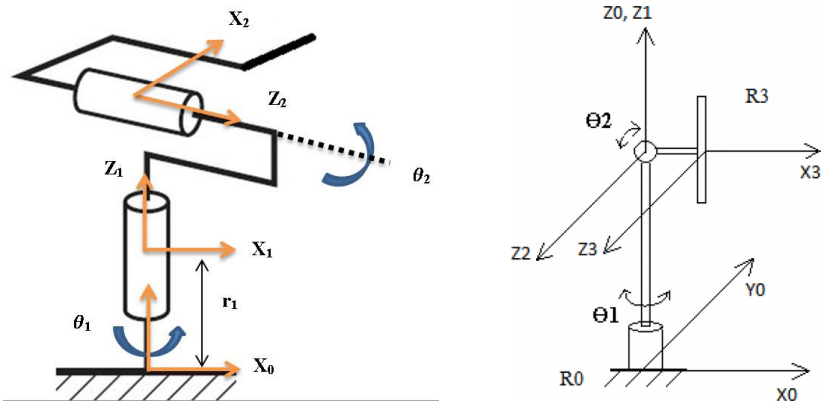
Reda และ Andreas [2] ได้รายงานวิธีการคำนวณหาตำแหน่งดวงอาทิตย์แบบอย่างละเอียดที่เรียกว่า SPA algorithm ซึ่งการคำนวณของ [2] จะสามารถคำนวณหาตำแหน่งดวงอาทิตย์ได้ในช่วง ค.ศ. -2000 ถึง 6000 และมีข้อผิดพลาดอยู่ที่ $\pm$0.0003 องศา เมื่อเปรียบเทียบวิธีการคำนวณของ Grena และ Reda แล้วพบว่าวิธีของ Grena จะใช้ขบวนการคำนวณน้อยกว่าของ Reda มาก แต่วิธีของ Reda มีความถูกต้องมากกว่า

Zang [3] และคณะ ได้กล่าวว่าโครงสร้างของ Heliostat โดยทั่วไปจะมีอยู่ด้วยกัน 2 โหมด คือ 1. โหมด Azimuth-Elevation 2. โหมด Spinning-Elevation และได้นำเสนอโครงสร้างแบบใหม่ที่เรียกว่า โหมด Latter และได้ทำการวิเคราะห์โครงสร้างทั้งในสภาวะ Statics และ Dynamics ด้วยวิธีไฟไนต์อิเลเมนต์



รูปที่ 2.1 Toroidal heliostat [3]

Chaib [4] ได้ออกแบบระบบระบบควบคุมของ Heliostat สำหรับสะท้อนแสงอาทิตย์ โดยคิดว่าเสาสะท้อนแสงอาทิตย์เป็นหุ่นยนต์ที่มีองศาอิสระเท่ากับ 2 และมีข้อต่อเป็นแบบ Revolute ทั้งหมด และทำการควบคุม Orientation ของหุ่นยนต์เสาสะท้อนแสงอาทิตย์ ส่วนโปรแกรมใช้การคำนวณการควบคุมนั้นได้ใช้ MATLAB มาช่วยและพีแอลซี Siemens S7-300 เป็นตัวสั่งงานและเก็บข้อมูล

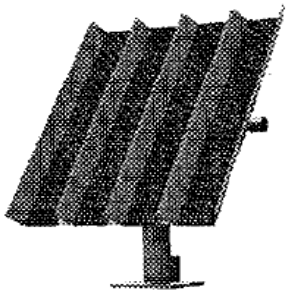


รูปที่ 2.2 หุ่นยนต์เสาสะท้อนแสงอาทิตย์ [4]

Rubio และคณะ [5] ได้นำเสนอการประยุกต์ใช้ระบบควบคุมสำหรับระบบติดตามดวงอาทิตย์ที่สามารถตามดวงอาทิตย์ด้วยความเที่ยงตรงสูง โดยปราศจากขั้นตอนของการติดตั้งที่มีความละเอียด โดยระบบควบคุมนี้เป็นระบบผสมระหว่างระบบติดตามดองอาทิตย์แบบวงเปิดที่ใช้แบบจำลองการเคลื่อนที่ของดวงอาทิตย์เป็นข้อมูลพื้นฐาน และระบบควบคุมแบบวงปิดที่ต้องใช้ตัวควบคุมป้อนกลับและพลศาสตร์ของระบบ โดยตัวควบคุมป้อนกลับนั้นเป็นแบบ PI controller โดยที่ระบบติดตามดอง

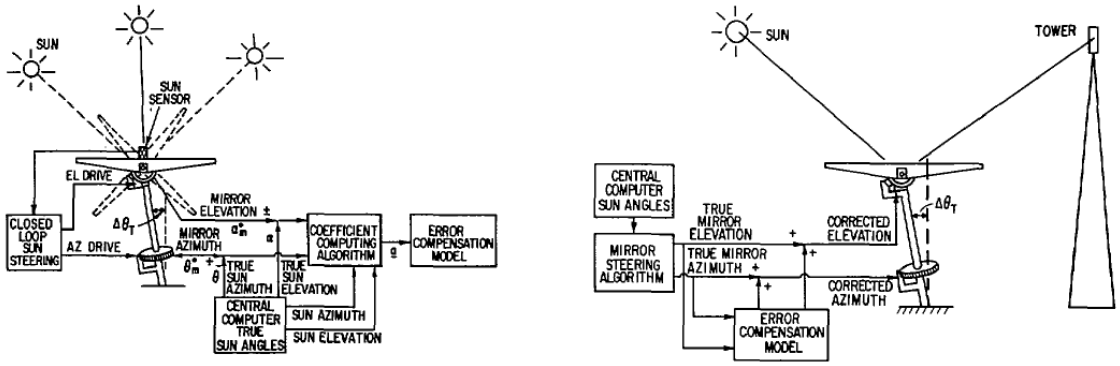
อาทิตย์แบบวงเปิดจะถูกใช้เมื่อตัวตรวจวัดดวงอาทิตย์ได้รับรังสีน้อยหรือตรวจหาดวงอาทิตย์ไม่พบ และในกรณีที่เริ่มทำงานในตอนเช้า และเลิกงานในตอนเย็น ส่วนระบบควบคุมแบบวงปิดก็จะถูกใช้ในกรณีปกติ คือตัวตรวจวัดดวงอาทิตย์ได้รับรังสีเพียงพอหรือตรวจหาดวงอาทิตย์พบ

Enrile และคณะ [6] ได้ออกแบบตัวต้นแบบของ Heliostat ที่เคลื่อนที่ได้ 2 แกนและสามารถสะท้อนแสงอาทิตย์ได้เป็นสองเท่า โดยหมุนรอบแกน Azimuth ได้ 0-240 องศา และแกน Elevation 0-90 องศา และมีความเที่ยงตรงที่ ±1 องศา ส่วนการติดตามดวงอาทิตย์นั้นใช้พื้นฐานของ Astronomical equations และควบคุมด้วยวงล้อมเปิด โดยใช้ PLC เป็นตัวควบคุม และใช้โมดูล CAN เน็ตเวิร์คของ PLC เป็นตัวสื่อสารข้อมูลจาก Heliostat หนึ่งไปยังอีก Heliostat หนึ่ง



รูปที่ 2.3 เสาสะท้อนแสงอาทิตย์ที่เคลื่อนที่ได้ 2 แกนของ Enrile และคณะ [6]

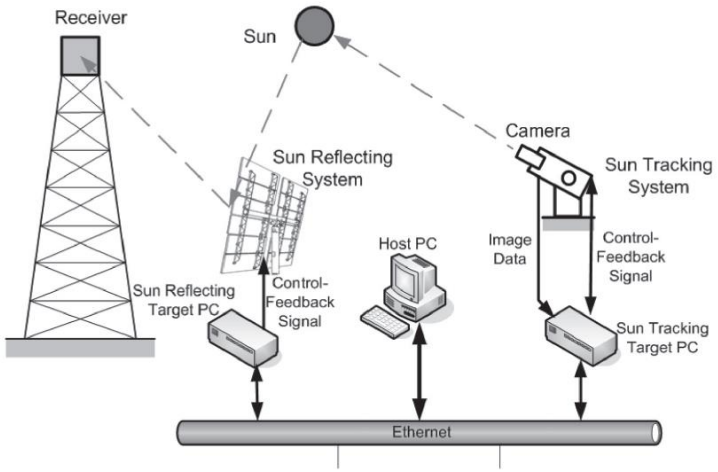
Baheti และ Scott [7] นำเสนอตัวควบคุมแบบปรับตั้งได้อัตโนมัติ (Self – calibrating) เพื่อใช้ในการลดความผิดพลาดของเสาสะท้อนแสงอาทิตย์ (Heliostat) จากการติดตั้งและการขับ โดยตัวควบคุมถูกออกแบบเป็น 2 โหมด โหมดแรกเรียกว่า โหมดปรับตั้ง (Calibration mode) โหมดนี้เสาสะท้อนแสงอาทิตย์จะถูกควบคุมด้วยระบบวงล้อมเปิด (Open – loop system) และแบบจำลองของค่าความผิดพลาดของเสาสะท้อนแสงอาทิตย์ จากการติดตั้งและการขับจะถูกคำนวณหาเอกลักษณ์โดยใช้ อัลกอริทึม Least – squares เพื่อหาพารามิเตอร์ของโมเดล โหมดที่สองเรียกว่า โหมดติดตาม (Tracking mode) โหมดนี้เสาสะท้อนแสงอาทิตย์จะถูกควบคุมด้วยระบบวงล้อมเปิด (Open loop - system) โดยใช้พารามิเตอร์ของโมเดลที่ได้จากโหมดแรกมาชดเชยค่าความผิดพลาดของเสาสะท้อนแสงอาทิตย์



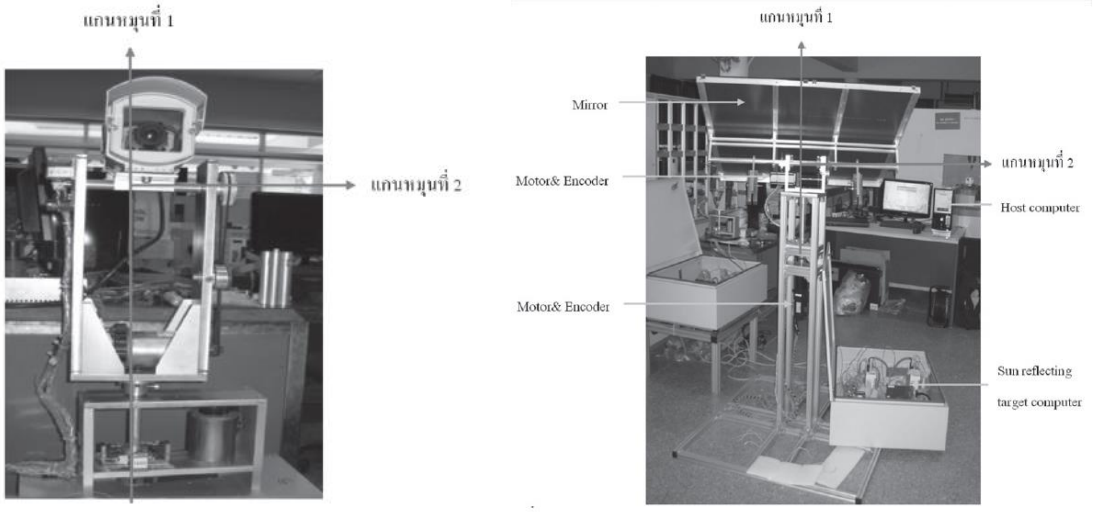
ก. Calibration mode ข. Tracking mode

รูปที่ 2.4 ตัวควบคุมแบบปรับตั้งได้อัตโนมัติของ Baheti และ Scott [7]

มนูศักดิ์ และ อธิราช [8] ได้นำเสนอระบบควบคุมการสะท้อนแสงอาทิตย์สำหรับระบบรับรังสีรวมกลางโดยระบบแบ่งออกเป็นสองส่วนคือระบบติดตามดวงอาทิตย์และระบบสะท้อแสงอาทิตย์ โดยที่โครงสร้างระบบติดตามดวงอาทิตย์มี 2 แกนหมุนและใช้กล้องดิจิตอลอุตสาหกรรมในการตรวจจับตำแหน่งของดวงอาทิตย์ ส่วนระบบสะท้อนแสงอาทิตย์มี2 แกนหมุนเช่นกันและติดตั้งแผ่นกระจกไว้ด้านบนโครงสร้างเพื่อเป็นตัวสะท้อนแสงอาทิตย์ ส่วนโปรแกรมควบคุมได้พัฒนาด้วยซอฟต์แวร์ LabVIEW และทำงานบนระบบปฏิบัติการเวลาจริงที่เรียกว่า Real-time target OS.



รูปที่ 2.5 แผนผังระบบควบคุมระบบติดตามและสะท้อนแสงอาทิตย์ [8]



ก. เสาติดตามดวงอาทิตย์ ข. เสาสะท้อนแสงอาทิตย์

รูปที่ 2.6 เสาติดตามดวงอาทิตย์และเสาสะท้อนแสงอาทิตย์ [8]

2.2 เทคโนโลยีผลิตไฟฟ้าด้วยระบบรวมแสงอาทิตย์

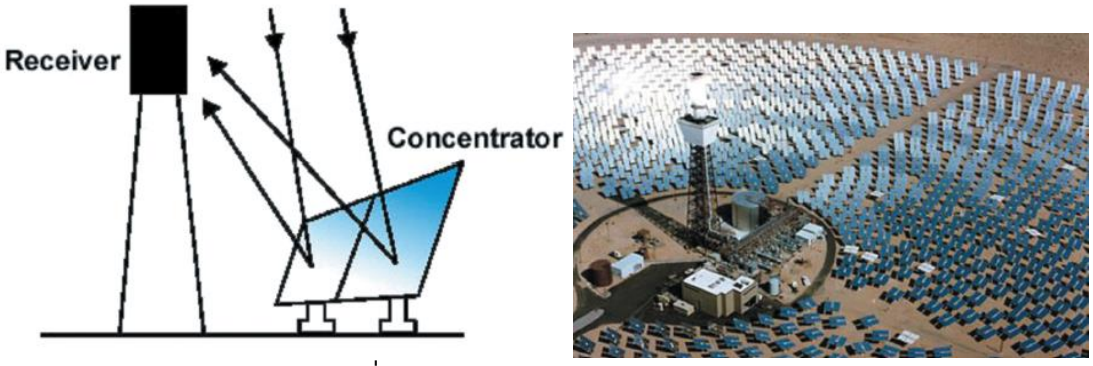
2.2.1 ระบบรางพาราโบลิก (Parabolic troughs)

ประกอบไปด้วยตัวรับแสงที่มีลักษณะเป็นรางยาวโค้งที่ติดตั้งบนระบบติด ตามดวงอาทิตย์แบบแกนเดียว ทำหน้าที่รวมแสงอาทิตย์และสะท้อนไปยังท่อที่ติดตั้งขนานกับแนวรางรวมแสง

รูปที่ 2.7 ระบบรางพาราโบลิก [12]

2.2.2 ระบบรับรังสีรวมกลาง (Central receivers)

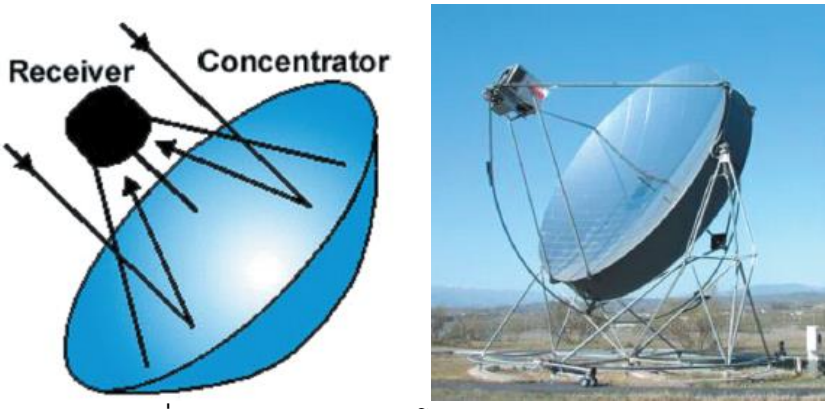
ประกอบไปด้วยตัวรับความร้อนที่ติดตั้งอยู่บนหอคอย และมีแผงกระจกจำนวนมากที่เรียกว่า เฮลิโอสแตท ล้อมรอบอยู่ โดยที่เฮลิโอสแตทจะหมุนเคลื่อนที่เพื่อที่จะสะท้อนแสงอาทิตย์ไปยังตัวรับความร้อนที่หอคอย



รูปที่ 2.8 ระบบรับรังสีรวมกลาง [12]

2.2.3 ระบบจานพาราโบรา (Parabolic dishes)

ประกอบไปด้วยตัวรวมแสงที่มีลักษณะเป็นจานรูปทรงพาราโบรา ที่มีจุดศูนย์รวมแสงเพื่อสะท้อนแสงอาทิตย์ไปยังตัวรับความร้อนที่ตั้งอยู่บนจุดศูนย์รวมของจาน โดยจานจะหมุนเคลื่อนที่ตามดวงอาทิตย์ตลอดเวลา

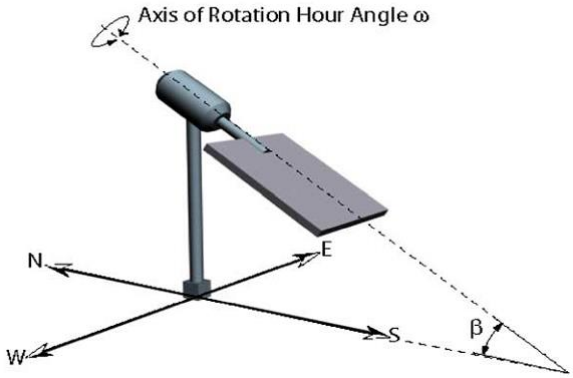


รูปที่ 2.9 ระบบจานพาราโบรา [12]

2.3 ลักษณะโครงสร้างของเสาสะท้อนหรือติดตามดวงอาทิตย์

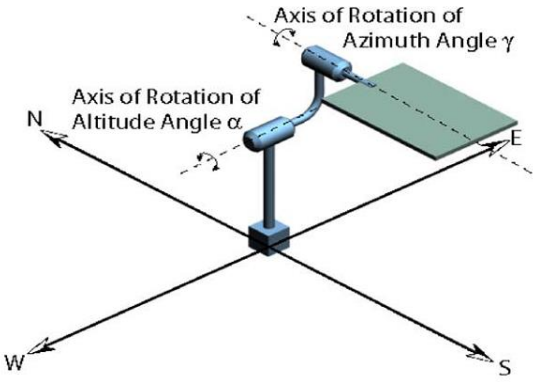
โครงสร้างของเสาสะท้อนหรือติดตามดวงอาทิตย์มีหลากหลายรูปแบบด้วยกัน อาทิเช่น

1. แบบหนึ่งแกน โครงสร้างแบบนี้จะหมุนสะท้อนหรือตามดวงอาทิตย์ได้เพียงแกนเดียว ซึ่งเป็นแกนแนวนอน



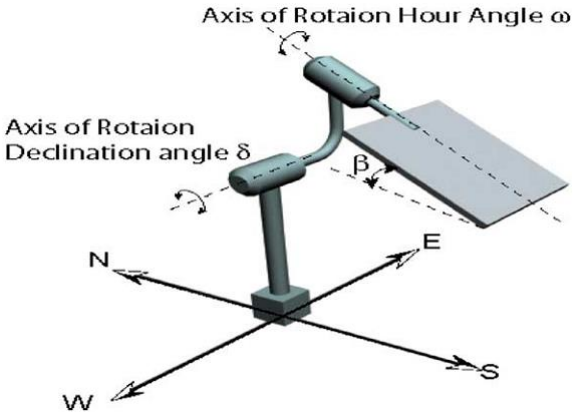
รูปที่ 2.10 One axis tracking

2. แบบสองแกนหรือ Azimuth elevation trackingโครงสร้างแบบนี้จะหมุนสะท้อนหรือตามดวงอาทิตย์ได้ 2 แกน ซึ่งได้แก่ Zenith axis และ Azimuth axis และใช้มอเตอร์จำนวนสองตัวในการขับเคลื่อนโครงสร้างให้เคลื่อนที่หรือหมุนรอบแกนทั้งสอง



รูปที่ 2.11 Azimuth elevation tracking

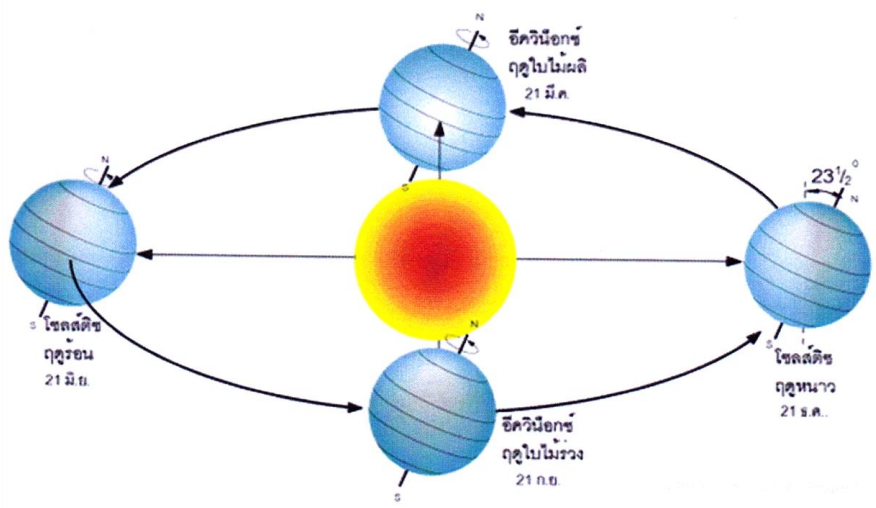
3. แบบ Equatorial tracking โครงสร้างแบบนี้จะหมุนสะท้อนหรือตามดวงอาทิตย์ได้ 2 แกนเหมือนกันกับแบบ Azimuth elevation tracking แต่ว่าแกนหมุนสุดท้ายจะเอียงเป็นมุม β



รูปที่ 2.12 Equatorial tracking

2.3 ตำแหน่งดวงอาทิตย์

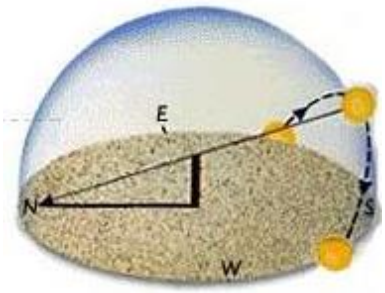
โลกโคจรรอบดวงอาทิตย์ในลักษณะที่เป็นวงรี แกนหมุนของโลกทำมุมเอียง 23.5 องศากับระนาบการโคจรรอบดวงอาทิตย์ โดยแกนของโลกจะชี้ไปตำแหน่งเดิมตลอดเวลา



รูปที่ 2.13 การโคจรของโลกรอบดวงอาทิตย์ [9]

ณ ตำแหน่งของโลกในวันที่ 21 กันยายน โลกจะเอียงด้านข้างให้กับดวงอาทิตย์ และแกนของโลกจะอยู่ในระนาบตั้งฉากกับรัศมีจากดวงอาทิตย์พอดี ทำให้แนวของแสงอาทิตย์จะอยู่บนระนาบของเส้นศูนย์สูตร ณ วันนี้ดวงอาทิตย์ขึ้นทางทิศตะวันออกและตกทางทิศตะวันตกพอดี ซีกโลกด้านเหนือและซีกโลกด้านใต้จะได้รับแสงอาทิตย์เท่ากัน และช่วงเวลากลางวันและกลางคืนเท่ากัน เท่ากับ 12 ชั่วโมง เรียกตำแหน่งนี้ว่า ศารทวิษุวัต (Autumnal equinox)

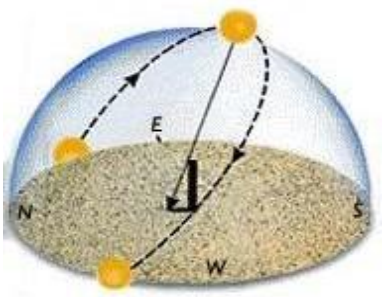
ณ ตำแหน่งของโลกในวันที่ 21 ธันวาคม โลกจะเอียงขั้วโลกใต้เข้าหาดวงอาทิตย์มากที่สุด ทำให้ตำแหน่งของดวงอาทิตย์ที่ปรากฏอยู่ต่ำสุด เรียกตำแหน่งนี้ว่า วินเตอร์โซลสตีส (Winter solstice) ช่วงนี้ประเทศทางซีกโลกใต้จะเป็นฤดูร้อน มีเวลากลางวันนานกว่ากลางคืน ส่วนประเทศทางซีกโลกเหนือจะเป็นฤดูหนาว มีเวลากลางวันสั้นกว่ากลางคืน การขึ้นและตกของดวงอาทิตย์จะค่อนไปทางใต้เล็กน้อย ดังรูปที่ 2.13



รูปที่ 2.14 การขึ้นและตกของดวงอาทิตย์ในวันที่ 21 ธันวาคม [10]

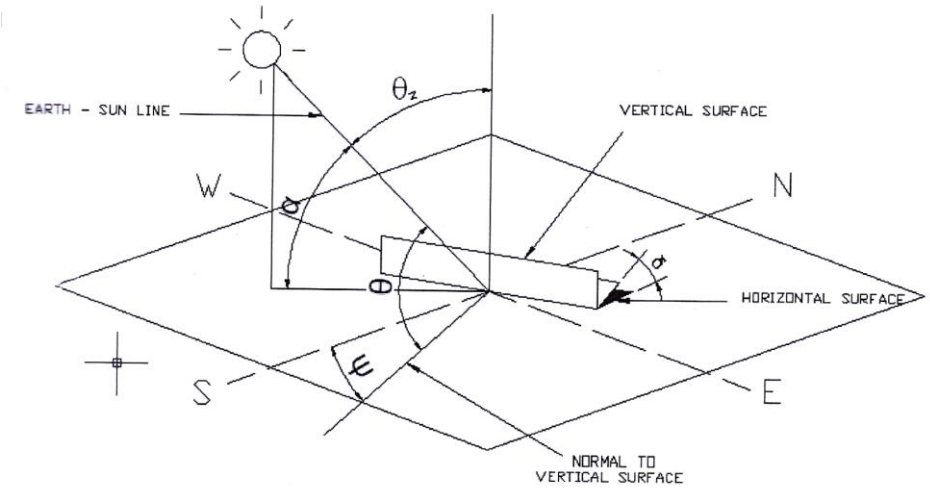
ณ ตำแหน่งของโลกในวันที่ 21 มีนาคม โลกจะเอียงด้านข้างให้กับดวงอาทิตย์ และแกนของโลกจะอยู่ในระนาบตั้งฉากกับรัศมีจากดวงอาทิตย์พอดี ทำให้แนวของแสงอาทิตย์จะอยู่บนระนาบของเส้นศูนย์สูตร ณ วันนี้ดวงอาทิตย์ขึ้นทางทิศตะวันออกและตกทางทิศตะวันตกพอดี ซีกโลกด้านเหนือและซีกโลกด้านใต้จะได้รับแสงอาทิตย์เท่ากัน และช่วงเวลากลางวันและกลางคืนเท่ากัน เท่ากับ 12 ชั่วโมง เรียกตำแหน่งนี้ว่า วสันตวิษุวัต (Vernal equinox)

ณ ตำแหน่งของโลกในวันที่ 21 มิถุนายน โลกจะเอียงขั้วโลกเหนือเข้าหาดวงอาทิตย์ ทำให้ตำแหน่งของดวงอาทิตย์บนท้องฟ้าอยู่สูงสุด คนที่อยู่บนเส้นละติจูดที่ 23.5 องศาเหนือจะเห็นดวงอาทิตย์อยู่ตรงศีรษะพอดีในตอนเที่ยงวัน เรียกตำแหน่งนี้ว่า ซัมเมอร์โซลสตีส (Summer solstice) ช่วงนี้ประเทศทางซีกโลกใต้จะเป็นฤดูหนาว มีเวลากลางวันสั้นกว่ากลางคืน ส่วนประเทศที่อยู่ทางซีกโลกเหนือจะเป็นฤดูร้อน มีเวลากลางวันนานกว่ากลางคืน การขึ้นและตกของดวงอาทิตย์จะค่อนไปทางเหนือเล็กน้อย ดังรูปที่ 2.15



รูปที่ 2.15 การขึ้นและตกของดวงอาทิตย์ในวันที่ 21 มิถุนายน [10]

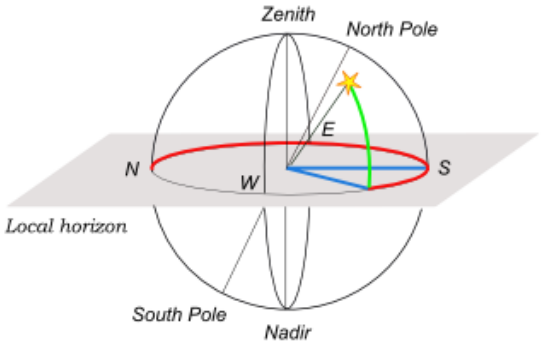
ความสัมพันธ์ระหว่างพื้นราบซึ่งวางในลักษณะใดๆ บนผิวโลกกับตำแหน่งของดวงอาทิตย์ ณ เวลาใดๆ สามารถแบ่งออกได้เป็นมุมต่างๆ ดังแสดงในรูปที่ 2.16



รูปที่ 2.16 มุมต่างๆ ที่มีความสัมพันธ์ระหว่างดวงอาทิตย์กับพื้นผิวโลก [10]

เมื่อ Ψ มุมอะซิมุธ (Azimuth angle) คือ ระยะทางเชิงมุมที่เบนไปจากเส้นแนว เหนือ – ใต้ เริ่มต้นทิศเหนือ มุมอะซิมุธมีค่าเป็น 0 องศา (ถ้าเบนไปทางทิศตะวันออกเป็นบวก เบนไปทางทิศตะวันตกเป็นลบ) ดังนั้นที่ทิศตะวันออกมุมอะซิมุธจะมีค่าเท่ากับ 90 องศา และที่ทิศตะวันตกมุมอะซิมุธจะมีค่าเท่ากับ -90 องศา

α มุมอัลติจูด (Altitude angle) หรือมุมเงย คือ ระยะทางเชิงมุมระหว่างดวงอาทิตย์กับพื้นราบ โดยมุมอัลติจูดจะมีค่าเป็น 0 องศาที่เส้นขอบฟ้า เมื่อเงยขึ้นจากเส้นขอบฟ้าให้มุมอัลติจูดมีค่าเป็นบวก มีค่าเท่ากับ 90 - θ_Z



รูปที่ 2.17 มุมอะซิมุธและมุมอัลติจูด [11]

δ มุมเดคคลิเนชั่น (Declination) คือ มุมระหว่างแนวลำแสงอาทิตย์ จะมีค่าเป็นบวกเมื่อวัดไปทางทิศเหนือ และมีค่าเป็นลบเมื่อวัดไปทางทิศใต้ มุมเดคคลิเนชั่นมีค่าเปลี่ยนไปทุกวันระหว่าง 23.45 องศา ถึง -23.45 องศา

2.3.1 ไทม์สเกล (Time scale)

- Universal time (UT) หรือ Greenwich civil time เป็นเวลาที่ใช้ในการรองรับการหมุนของโลกและเริ่มนับ 0.00 น. ตอนเที่ยงคืน
- International atomic time (TAI) เป็นระยะเวลาของ System International Second (SI-second)
- Coordinated Universal Time (UTC) เป็นพื้นฐานของสัญญาณวิทยุและระบบเวลาตามกฎหมาย
- Terrestrial Dynamical หรือ Terrestrial Time (TDT or TT) เป็นไทม์สเกลของ Ephemerides สำหรับการสังเกตุจากบนพื้นผิวโลก

สมการด้านล่างนี้จะแสดงความสัมพันธ์ระหว่างไทม์สเกลด้านบน ในหน่วย วินาที

$$TT = TAI + 32.184 \tag{2.1}$$

$$UT = TT - \Delta T \tag{2.2}$$

โดยที่ ΔT คือค่าแตกต่างระหว่างเวลาการหมุนของโลกกับ Terrestrial Time (TT)

$$UT = UT1 = UTC + \Delta UT1 \tag{2.3}$$

โดยที่ $\Delta UT1$ คือแฟรคชั่นของวินาที จะเป็นมีค่าเป็นบวกหรือลบก็ได้

2.3.3 ขั้นตอนการคำนวณหาตำแหน่งดวงอาทิตย์ด้วย SPA อัลกอลิธึม

1. คำนวณหาค่า Julian Day, Julian Ephemeris Day, Century และ Millennium

1.1 คำนวณหา Julian Day (JD)

$$JD = INT\big(365.25 * (Y + 4716)\big) + INT\big(30.6001 * (M + 1)\big) + D + B - 1524.5 \tag{2.4}$$

โดยที่ INT คือ ค่าจำนวนเต็มของเทอมที่ถูกคำนวณ เช่น INT(8.3) = 8 หรือ INT(-3.7) = -3

Y คือ ปี ค.ศ. เช่น 2014, 2015 เป็นต้น

M คือ เดือนของปี เช่น M=1 เมื่อเดือนเป็นเดือนมกราคม เป็นต้น โดยที่ ถ้า M>2 จะทำให้ค่า Y และ M ไม่เปลี่ยนแปลง แต่ว่าค่า M=1 หรือ 2 ค่า Y และ M จะต้องคำนวณจาก Y=Y-1 และ M=M+12

D คือ วันของเดือน ในหน่วย Decimal time เช่น วันที่ 2 ของเดือนที่เวลา 12:30:30 UT จะได้ค่า D=2.521180556

B คือ จะมีค่าเท่ากับ 0 สำหรับ Julian calendar และมีค่าเท่ากับ 2–A+INT(A/4) สำหรับ Gregorian calendar และ A=INT(Y/100)

1.2 คำนวณหา Julian Ephemeris Day (JDE)

$$JDE = JD + \frac{\Delta T}{86400} \quad (2.5)$$

1.3 คำนวณหา Julian century (JC) และ Julian Ephemeris Century (JCE)

$$JC = \frac{JD-2451545}{36525} \quad (2.6)$$

$$JCE = \frac{JDE-2451545}{36525} \quad (2.7)$$

1.4 คำนวณหา Julian Ephemeris Millennium (JME)

$$JME = \frac{JCE}{10} \quad (2.8)$$

2. คำนวณหาค่า Earth heliocentric longitude, latitude และ radius vector (L, B และ R)

2.1 คำนวณหาเทอม $L0_i$ (in radians)

$$L0_i = A_i cos(B_i + C_i \times JME) \quad (2.9)$$

โดยที่ i คือ ลำดับของแถว สำหรับเทอม $L0$ ในตารางที่ ก.1

A_i, B_i และ C_i คือ ค่าในแถวที่ i และคอลัมน์ที่ A, B และ C ในตารางที่ ก.1

2.2 คำนวณหาเทอม $L0$ (in radians)

$$L0 = \sum_{i=0}^{n} L0_i \quad (2.10)$$

โดยที่ n คือจำนวนแถวของเทอม $L0$ ในตารางที่ ก.1

2.3 คำนวณหาเทอม $L1, L2, L3, L4$ และ $L5$ โดยใช้สมการที่ 2.9 และ 2.10 แต่เปลี่ยนจาก 0 เป็น 1, 2, 3, 4 และ 5 โดยใช้ค่าต่างๆ ในตารางที่ ก.1 หน่วยที่ได้จะเป็นเรเดียน

2.4 คำนวณหา Earth heliocentric longitude, L (in radians)

$$L = \frac{L0+L1\times JME+L2\times JME^2+L3\times JME^3+L4\times JME^4+L5\times JME^5}{10^8} \quad (2.11)$$

2.5 คำนวณหา L (in degrees)

$$L\ (in\ Degrees) = \frac{L\ (in\ Radian)\times 180}{\pi} \quad (2.12)$$

โดยที่ π มีค่าประมาณ 3.1415926535898

2.6 จำกัดค่า L ให้อยู่ในช่วง 0 ถึง 360 องศา โดยการนำเอาค่า L มาหารด้วย 360 จากนั้นเก็บค่าเป็นตัวแปร F และถ้า L เป็นบวก ค่า $L = 360F$ แต่ถ้า L ลบ ค่า $L = 360 - 360F$

2.7 คำนวณหาค่า Earth heliocentric latitude, B (in degrees) โดยใช้ตารางที่ ก.1 และขั้นตอนที่ 2.1 ถึง 2.5 โดยการแทนตัวแปร L ไปเป็น B ในทุกๆ สมการ

2.8 คำนวณหา Earth radius vector, R (in Astronomical units, AU) โดยใช้ตารางที่ ก.1 และขั้นตอนที่ 2.1 ถึง 2.5 โดยการแทนตัวแปร L ไปเป็น R ในทุกๆ สมการ

3. คำนวณหาค่า Geocentric longitude และ latitude, $\emptyset$ และ β

3.1 คำนวณหา Geocentric longitude, $\emptyset$ (in degrees)

$$\emptyset = L + 180 \quad (2.13)$$

3.2 จำกัดค่า $\emptyset$ ให้อยู่ในช่วง 0 ถึง 360 องศา โดยการทำตามขั้นตอนที่ 2.6

3.3 คำนวณหา Geocentric latitude, β (in degrees)

$$\beta = -B \quad (2.14)$$

4. คำนวณหาค่า Nutation in longitude and obliquity, $\Delta\vartheta$ และ $\Delta\varepsilon$

4.1 คำนวณหา Elongation เฉลี่ยของดวงจันทร์จากดวงอาทิตย์, X_0 (in degrees)

$$X_0 = 297.85036 + 445267.111480 \times JCE - 0.0019142 \times JCE^2 + \frac{JCE^3}{189474} \quad (2.15)$$

4.2 คำนวณหา Anomaly เฉลี่ยของดวงอาทิตย์ (โลก), X_1 (in degrees)

$$X_1 = 357.52778 + 35999.05034 \times JCE - 0.0001603 \times JCE^2 - \frac{JCE^3}{300000} \quad (2.16)$$

4.3 คำนวณหา Anomaly เฉลี่ยของดวงจันทร์, X_2 (in degrees)

$$X_2 = 134.96298 + 477198.867398 \times JCE + 0.0086972 \times JCE^2 + \frac{JCE^3}{56250} \quad (2.17)$$

4.4 คำนวณหา Moon's argument of latitude, X_3 (in degrees)

$$X_3 = 93.27191 + 483202.017538 \times JCE - 0.0036825 \times JCE^2 + \frac{JCE^3}{327270} \quad (2.18)$$

4.5 คำนวณหา Longitude of the ascending node of the moon's mean orbit on the ecliptic, วัดจาก Equinox เฉลี่ย ของ date, X_4 (in degrees)

$$X_4 = 125.04452 - 1934.136261 \times JCE + 0.0020708 \times JCE^2 + \frac{JCE^3}{450000} \quad (2.19)$$

4.6 สำหรับแถวแต่ละแถวในตารางที่ ก.1, คำนวณหาเทอม $\Delta\vartheta_i$ และ $\Delta\varepsilon_i$ ใน 0.001 ของ arc seconds

$$\Delta\vartheta_i = (a_i + b_i \times JCE) \times sin\left(\sum_{j=0}^{4} X_j Y_{i,j}\right) \quad (2.20)$$

$$\Delta\varepsilon_i = (c_i + d_i \times JCE) \times cos\left(\sum_{j=0}^{4} X_j Y_{i,j}\right) \quad (2.21)$$

โดยที่ a_i, b_i, c_i และ d_i เป็นค่าที่อยู่ในแถวที่ i และคอลัมน์ที่ a, b, c และ d ในตารางที่ ก.2
X_j เป็นค่า X ที่ j คำนวณโดยใช้สมการที่ 3.15 ถึง 3.19
$Y_{i,j}$ เป็นค่าในตารางที่ ก.2 ในแถวที่ i และคอลัมน์ Y ที่ j

4.7 คำนวณหา Nutation in longitude, $\Delta\vartheta$ (in degrees)

$$\Delta\vartheta = \frac{\sum_{i=0}^{n} \Delta\vartheta_i}{36000000} \quad (2.22)$$

โดยที่ n เป็นจำนวนของแถวในตารางที่ ก.2 (n=63)

4.8 คำนวณหา Nutation in obliquity, $\Delta\varepsilon$ (in degrees)

$$\Delta\varepsilon = \frac{\sum_{i=0}^{n} \Delta\varepsilon_i}{36000000} \tag{2.23}$$

5. คำนวณหาค่า True obliquity of the ecliptic, ε (in degrees)

5.1 คำนวณหา Mean obliquity of the ecliptic, ε_0 ใน arc seconds

$$\varepsilon_0 = 84381.448 - 4680.93U - 1.55U^2 + 1999.25U^3 - 51.38U^4 - 249.67U^5 - 39.05U^6 + 7.12U^7 + 27.87U^8 + 5.79U^9 + 2.45U^{10} \tag{2.24}$$

โดยที่ $U = JME/10$

5.2 คำนวณหา True obliquity of the ecliptic, ε (in degrees)

$$\varepsilon = \frac{\varepsilon_0}{3600} + \Delta\varepsilon \tag{2.25}$$

6. คำนวณหาค่า Aberration correction, $\Delta\tau$ (in degrees)

$$\Delta\tau = -\frac{20.4898}{3600R} \tag{2.26}$$

7. คำนวณหาค่า Apparent sun longitude, λ (in degrees)

$$\lambda = \emptyset + \Delta\vartheta + \Delta\tau \tag{2.27}$$

8. คำนวณหาค่า Apparent sidereal time at Greenwich at any given time, v (in degrees)

8.1 คำนวณหา Mean sidereal time at Greenwich, v_0 (in degrees)

$$v_0 = 280.46061837 + 360.98564736629 \times (JD - 2451545) + 0.000387933 \times JC^2 - \frac{JC^3}{38710000} \tag{2.28}$$

8.2 จำกัดค่า v_0 ให้อยู่ในช่วง 0 ถึง 360 องศา โดยการทำตามขั้นตอนที่ 2.6

8.3 คำนวณหา Apparent sidereal time at Greenwich, v (in degrees)

$$v = v_0 + \Delta\vartheta \cos\varepsilon \tag{2.29}$$

9. คำนวณหาค่า Geocentric sun right ascension, α (in degrees)

9.1 คำนวณหา Geocentric sun right ascension, α (in radians)

$$\alpha = Arctan2\left(\frac{\sin\lambda\cos\varepsilon - \tan\beta\sin\varepsilon}{\cos\lambda}\right) \tag{2.30}$$

โดยที่ Arctan2 เป็นฟังก์ชัน arctangent

9.2 คำนวณหา α ในหน่วยดีกรี โดยใช้วิธีการตามสมการที่ 2.12

10. คำนวณหาค่า Geocentric sun declination, δ (in degrees)

$$\delta = \frac{180 \times Arcsin(\sin\beta\cos\varepsilon + \cos\beta\sin\varepsilon\sin\lambda)}{\pi} \tag{2.31}$$

โดยที่ค่า δ เป็นบวกหรือลบ ขึ้นกับดวงอาทิตย์อยู่ทางทิศเหนือหรือใต้ ของ celestial equator ตามลำดับ

11. คำนวณหาค่า Observer local hour angle, H (in degrees)

$$H = v + \sigma - \alpha \tag{2.32}$$

โดยที่ σ คือค่า Observer geographical longitude จะมีค่าเป็นบวกหรือลบ ขึ้นอยู่กับว่าอยู่ทางทิศตะวันออกหรือตะวันตกของ Greenwich ตามลำดับ และค่า H จะต้องถูจำกัดช่วงให้อยู่ในช่วง 0 ถึง 360 องศา ตามขั้นตอนที่ 2.6

12. คำนวณหาค่า Topocentric sun right ascension, α' (in degrees)

12.1 คำนวณหา Equatorial horizontal parallax of the sun, ξ (in degrees)

$$\xi = \frac{8.794}{3600R} \tag{2.33}$$

โดยที่ค่า R คำนวณได้จากขั้นตอนที่ 2.8

12.2 คำนวณหาเทอม u (in radians)

$$u = Arc\ tan\ (0.99664719 \tan \varphi) \quad (2.34)$$

โดยที่ φ คือค่า Observer geographical latitude จะมีค่าเป็นบวกหรือลบ ขึ้นอยู่กับว่าอยู่ทางทิศเหนือหรือใต้ของ Equator ตามลำดับ

12.3 คำนวณหาเทอม x

$$x = \cos u + \frac{E}{6378140} \cos \varphi \quad (2.35)$$

โดยที่ E คือค่า Observer elevation (in meters)

12.4 คำนวณหาเทอม y

$$y = 0.99664719 \sin u + \frac{E}{6378140} \sin \varphi \quad (2.36)$$

12.5 คำนวณหา Parallax in the sun right ascension, $\Delta\alpha$ (in degrees)

$$\Delta\alpha = Arc\ tan2\ \left(\frac{-x \sin \xi \sin H}{\cos \delta - x \sin \xi \cos H}\right) \times \frac{180}{\pi} \quad (2.37)$$

12.6 คำนวณหา Topocentric sun right ascension, α' (in degrees)

$$\alpha' = \alpha + \Delta\alpha \quad (2.38)$$

12.7 คำนวณหา topocentric sun declination, δ' (in degrees)

$$\delta' = Arc\ tan2\ \left(\frac{(\sin \delta - y \sin \xi) \cos \Delta\alpha}{\cos \delta - x \sin \xi \cos H}\right) \quad (2.39)$$

13. คำนวณหาค่า Topocentric local hour angle, H' (in degrees)

$$H' = H - \Delta\alpha \quad (2.40)$$

14. คำนวณหาค่า Topocentric zenith angle, θ (in degrees)

14.1 คำนวณหา Topocentric elevation angle without atmospheric refraction correction, e_0 (in degrees)

$$e_0 = Arc\ sin\ (\sin\varphi \sin\delta' + \cos\varphi \cos\delta' \cos H') \times \frac{180}{\pi} \quad (2.41)$$

14.2 คำนวณหา Atmospheric refraction correction, Δe (in degrees)

$$\Delta e = \frac{P}{1010} \times \frac{283}{273+T} \times \frac{1.02}{60 \tan\left(e_0 + \frac{10.3}{e_0+5.11}\right)} \quad (2.42)$$

โดยที่ P คือค่าความดันเฉลี่ยของพื้นที่ในหนึ่งปี มีหน่วย milli Bar
T คือค่าอุณหภูมิเฉลี่ยของพื้นที่ในหนึ่งปี มีหน่วย องศาเซลเซียส

14.3 คำนวณหา Topocentric elevation angle, e (in degrees)

$$e = e_0 + \Delta e \quad (2.43)$$

14.4 คำนวณหา Topocentric zenith angle, θ (in degrees)

$$\theta = 90 - e \quad (2.44)$$

15. คำนวณหาค่า Topocentric azimuth angle, Φ (in degrees)

15.1 คำนวณหา Topocentric astronomers azimuth angle, Γ (in degrees)

$$\Gamma = Arc\ tan2\ \left(\frac{\sin H'}{\cos H' \sin\varphi - \tan\delta' \cos\varphi}\right) \times \frac{180}{\pi} \quad (2.45)$$

ค่า Γ จะต้องถูกจำกัดช่วงให้อยู่ในช่วง 0 ถึง 360 องศา ตามขั้นตอนที่ 2.6 ค่า Γ จะถูกวัดไปทางทิศตะวันตกเริ่มจากทิศใต้

15.2 คำนวณหา Topocentric azimuth angle, Φ (in degrees)

$$\Phi = \Gamma + 180 \quad (2.46)$$

ค่า Φ จะต้องถูกจำกัดช่วงให้อยู่ในช่วง 0 ถึง 360 องศา ตามขั้นตอนที่ 2.6 ค่า Φ จะถูกวัดไปทางทิศตะวันออกเริ่มจากทิศเหนือ

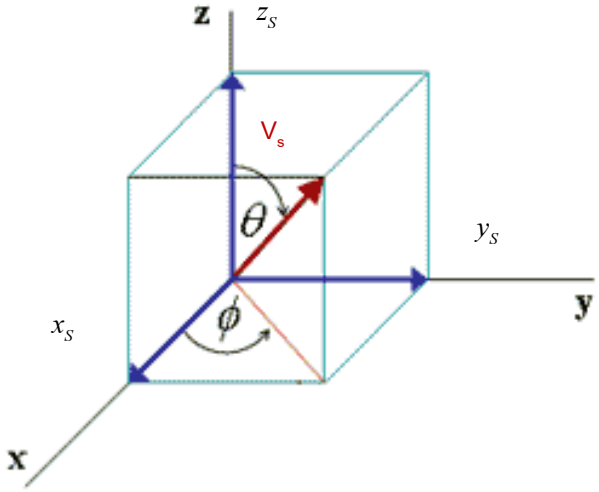
16. คำนวณหาค่า Incidence angle for a surface oriented in any direction, I (in degrees)

$$I = Arccos\ (\cos\theta\cos\omega + \sin\omega\ \sin\theta\cos(\Gamma - \gamma)) \times \frac{180}{\pi} \quad (2.47)$$

โดยที่ ω เป็นค่าความชันของพื้นผิวโดยวัดจากระนาบแนวนอน ส่วน γ เป็นค่า Surface azimuth rotation angle โดยวัดจากทิศใต้ไปจนถึงเส้นโปรเจคชันของเส้นตั้งฉากพื้นผิวบนระนาบแนวนอน

2.5 การหามุมหมุนของเสาสะท้อนแสงอาทิตย์

ในการควบคุมระบบสะท้อนแสงอาทิตย์ จำเป็นต้องหาเวกเตอร์ตำแหน่งดวงอาทิตย์ (Sun - position vector) โดยอาศัยการแปลงมุมซีนิซและมุมอซิมุธของดวงอาทิตย์ ให้เป็นเวกเตอร์ตำแหน่งดวงอาทิตย์ สามารถหาได้ดังนี้



รูปที่ 2.18 เวกเตอร์ตำแหน่งดวงอาทิตย์ [8]

$$\mathbf{V_S} = x_S\mathbf{i} + y_S\mathbf{j} + z_S\mathbf{k} \quad (2.48)$$

$$|\mathbf{V_S}| = \sqrt{(x_S)^2 + (y_S)^2 + (z_S)^2} \quad (2.49)$$

เมื่อ $\mathbf{V_S}$ คือ เวกเตอร์ตำแหน่งดวงอาทิตย์

$|\mathbf{V_S}|$ คือ ขนาดของเวกเตอร์ตำแหน่งดวงอาทิตย์

x_S คือ เวกเตอร์ตำแหน่งดวงอาทิตย์ ในแกน x สามารถหาได้จาก

$$x_S = \sin\theta\cos\emptyset \quad (2.50)$$

y_S คือ เวกเตอร์ตำแหน่งดวงอาทิตย์ ในแกน y สามารถหาได้จาก

$$y_S = \sin\theta \sin\emptyset \tag{2.51}$$

z_S คือ เวกเตอร์ตำแหน่งดวงอาทิตย์ ในแกน z สามารถหาได้จาก

$$z_S = \cos\theta \tag{2.52}$$

โดยที่ θ คือ มุมซีนิช
$\emptyset$ คือ มุมอซิมุธ

จากสมการที่ (2.48) และ (2.49) สามารถหาเวกเตอร์หนึ่งหน่วย (Unit Vector) ได้ดังนี้

$$\mathbf{n_S} = \frac{\mathbf{V_S}}{|\mathbf{V_S}|} = n_{Sx}\mathbf{i} + n_{Sy}\mathbf{j} + n_{Sz}\mathbf{k} \tag{2.53}$$

เมื่อ n_{Sx} คือ เวกเตอร์ตั้งฉาก (Normal vector) ของดวงอาทิตย์ในแกน x สามารถหาได้จาก

$$n_{Sx} = \frac{x_S}{\sqrt{(x_S)^2+(y_S)^2+(z_S)^2}} \tag{2.54}$$

n_{Sy} คือ เวกเตอร์ตั้งฉากของดวงอาทิตย์ในแกน y สามารถหาได้จาก

$$n_{Sy} = \frac{y_S}{\sqrt{(x_S)^2+(y_S)^2+(z_S)^2}} \tag{2.55}$$

n_{Sz} คือ เวกเตอร์ตั้งฉากของดวงอาทิตย์ในแกน z สามารถหาได้จาก

$$n_{sz} = \frac{z_s}{\sqrt{(x_S)^2+(y_S)^2+(z_S)^2}} \tag{2.56}$$

เวกเตอร์ตำแหน่งของเป้าหมาย (Target position vector) คือ ตำแหน่งเป้าหมายบริเวณที่ต้องการให้แสงอาทิตย์สะท้อนไปตกบริเวณนั้น สามารถหาได้โดยการวัดโดยตรง จะได้ดังนี้

$$\mathbf{V_T} = x_T\mathbf{i} + y_T\mathbf{j} + z_T\mathbf{k} \tag{2.57}$$

$$|\mathbf{V_T}| = \sqrt{(x_T)^2 + (y_T)^2 + (z_T)^2} \tag{2.58}$$

เมื่อ $\mathbf{V_T}$ คือ เวกเตอร์ตำแหน่งเป้าหมาย

$|\mathbf{V_S}|$ คือ ขนาดของเวกเตอร์เป้าหมาย

x_T คือ เวกเตอร์ตำแหน่งเป้าหมายในแกน x

y_T คือ เวกเตอร์ตำแหน่งเป้าหมายในแกน y

z_T คือ เวกเตอร์ตำแหน่งเป้าหมายในแกน z

จากสมการที่ (2.57) และ (2.58) สามารถหาเวกเตอร์หนึ่งหน่วย ได้ดังนี้

$$\mathbf{n_T} = \frac{\mathbf{V_T}}{|\mathbf{V_T}|} = n_{Tx}\mathbf{i} + n_{Ty}\mathbf{j} + n_{Tz}\mathbf{k} \tag{2.59}$$

เมื่อ n_{Tx} คือ เวกเตอร์ตั้งฉากของเป้าหมายในแกน x สามารถหาได้จาก

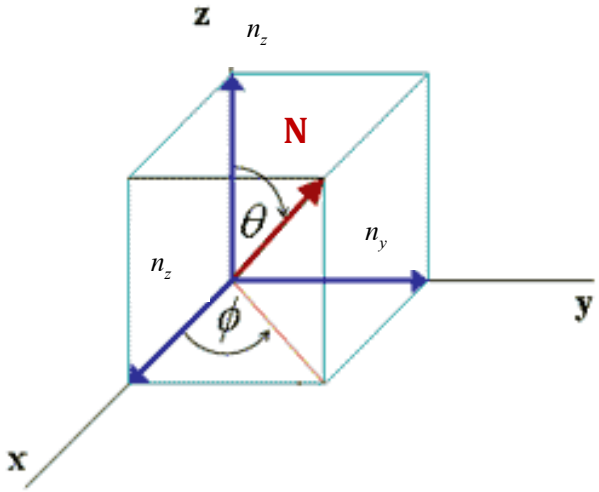
$$n_{Tx} = \frac{x_T}{\sqrt{(x_T)^2+(y_T)^2+(z_T)^2}} \tag{2.60}$$

n_{Ty} คือ เวกเตอร์ตั้งฉากของเป้าหมายในแกน y สามารถหาได้จาก

$$n_{Ty} = \frac{y_T}{\sqrt{(x_T)^2+(y_T)^2+(z_T)^2}} \tag{2.61}$$

n_{Tz} คือ เวกเตอร์ตั้งฉากของเป้าหมายในแกน z สามารถหาได้จาก

$$n_{Tz} = \frac{z_T}{\sqrt{(x_T)^2+(y_T)^2+(z_T)^2}} \tag{2.62}$$



รูปที่ 2.19 เวกเตอร์ตั้งฉากของระบบสะท้อนแสงอาทิตย์ [8]

การหามุมในการหมุนของระบบสะท้อนแสงอาทิตย์ สามารถหาได้โดยอาศัยความสัมพันธ์ระหว่างเวกเตอร์ตำแหน่งดวงอาทิตย์และเวกเตอร์ตำแหน่งของเป้าหมาย โดยสามารถหาเวกเตอร์ตั้งฉากของระบบสะท้อนแสงอาทิตย์ (**N**) ได้ดังนี้

$$\mathbf{N} = \frac{\mathbf{n_S}+\mathbf{n_T}}{|\mathbf{n_S}+\mathbf{n_T}|} = n_x\mathbf{i} + n_y\mathbf{j} + n_z\mathbf{k} \tag{2.63}$$

เมื่อ n_x คือ เวกเตอร์ตั้งฉากของระบบสะท้อนแสงอาทิตย์ในแกน x สามารถหาได้จาก

$$n_x = \frac{(n_{Sx}+n_{Tx})}{\sqrt{(n_{Sx}+n_{Tx})^2+(n_{Sy}+n_{Ty})^2+(n_{Sz}+n_{Tz})^2}} \tag{2.64}$$

n_y คือ เวกเตอร์ตั้งฉากของระบบสะท้อนแสงอาทิตย์ในแกน y สามารถหาได้จาก

$$n_y = \frac{(n_{Sy}+n_{Ty})}{\sqrt{(n_{Sx}+n_{Tx})^2+(n_{Sy}+n_{Ty})^2+(n_{Sz}+n_{Tz})^2}} \tag{2.65}$$

n_z คือ เวกเตอร์ตั้งฉากของระบบสะท้อนแสงอาทิตย์ในแกน z สามารถหาได้จาก

$$n_z = \frac{(n_{Sz}+n_{Tz})}{\sqrt{(n_{Sx}+n_{Tx})^2+(n_{Sy}+n_{Ty})^2+(n_{Sz}+n_{Tz})^2}} \tag{2.66}$$

จากรูปที่ 2.18 จะสามารถหามุมในการหมุนของเสาสะท้อนแสงอาทิตย์ได้ดังนี้

$$\theta = \cos^{-1}(n_z) \tag{2.67}$$

$$\emptyset = \sin^{-1}\left[\frac{n_y}{\sqrt{(n_x)^2+(n_y)^2}}\right] \tag{2.68}$$

เมื่อ

θ คือ มุมซีนิชในการหมุนของเสาสะท้อนแสงอาทิตย์ (มุมอัลติจูดเท่ากับ 90 - θ)

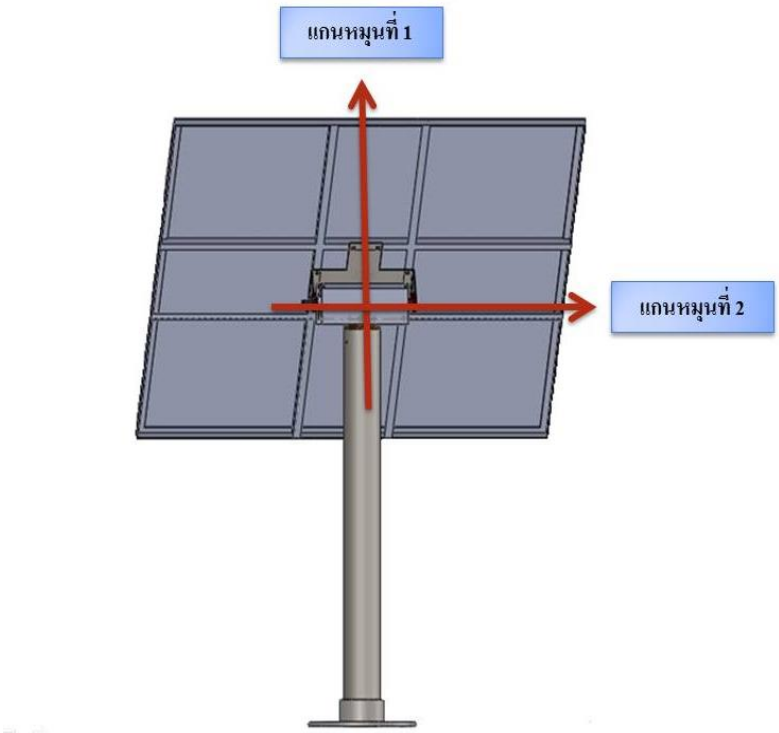
$\emptyset$ คือ มุมอซิมุธในการหมุนของเสาสะท้อนแสงอาทิตย์

บทที่ 3
โครงสร้างของระบบ

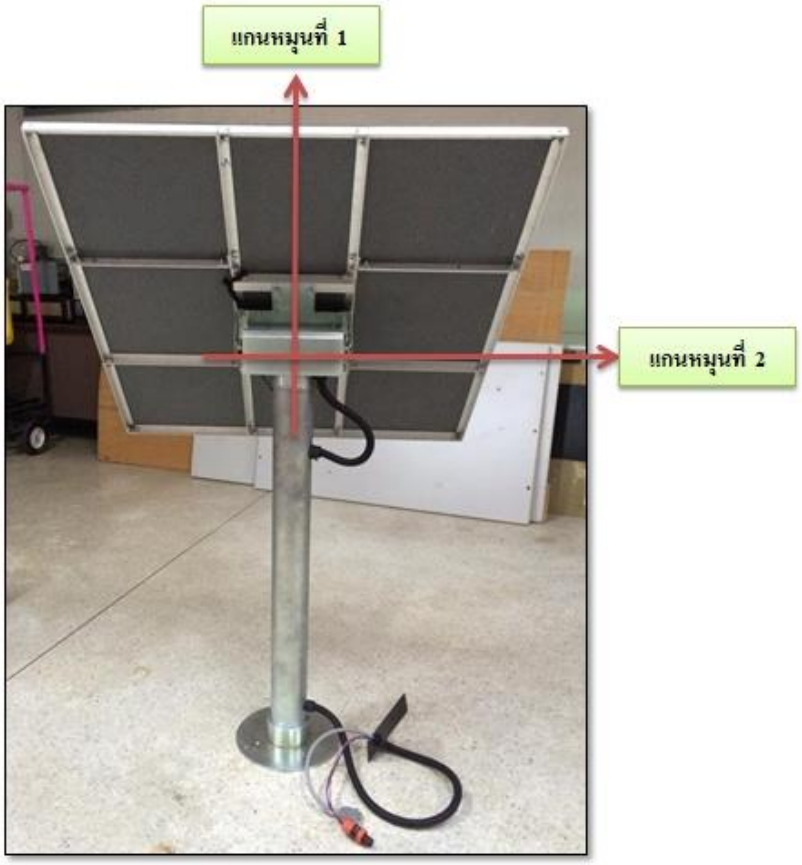
บทนี้จะเป็นการอธิบายถึงรายละเอียดของโครงสร้างของเสาสะท้อนแสงอาทิยต์

3.1 เสาสะท้อนแสงอาทิตย์

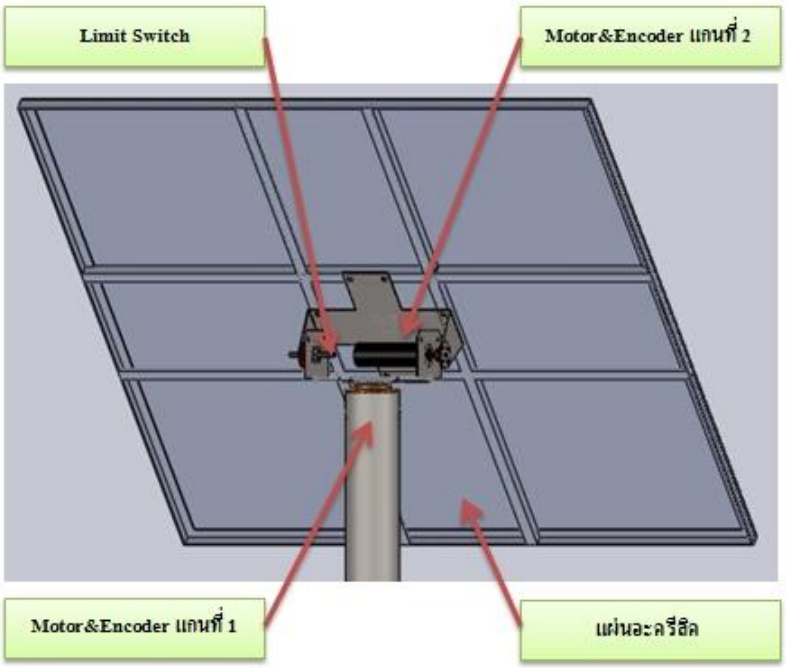
เสาสะท้อนแสงอาทิตย์ที่ได้ออกแบบจะมีแกนหมุนในการเคลื่อนที่ได้ 2 แกน และใช้ดีซีเซอร์โวมอเตอร์ขนาด 24 VDC โดยแกนหมุนทั้งสองสามารถหมุนในขอบเขตการเคลื่อนที่ 180 องศา และใช้ลิมิตสวิทช์เป็นตัวกำหนดขอบเขตการหมุน ในการออกแบบมอเตอร์ที่ใช้จะให้มีขนาดเล็กลงตามน้ำหนักและขนาดของโครงสร้างที่เล็กลงซึ่งบนเพลาแกนหมุนที่ 2 นี้จะติดตั้งแผ่นอะครีลิคไว้ด้านบนเพื่อเป็นตัวสะท้อนแสงอาทิตย์ โดยจะมีน้ำหนักเบากว่าระบบเดิมที่ใช้กระจกในการสะท้อนแสง ดังรูปที่ 3.1



รูปที่ 3.1 การออกแบบโครงสร้างระบบสะท้อนแสงอาทิตย์



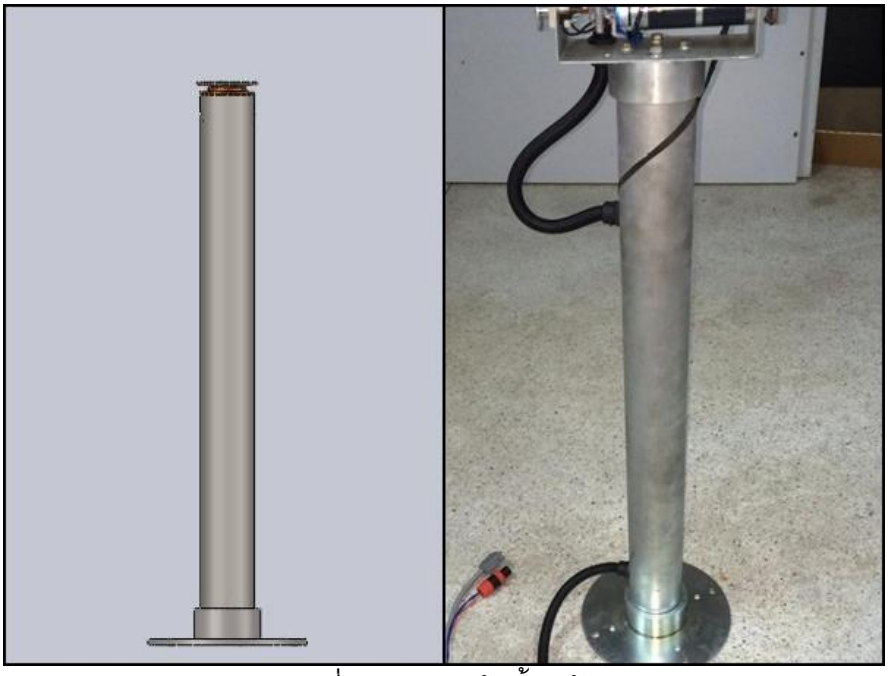
รูปที่ 3.2 โครงสร้างระบบสะท้อนแสงอาทิตย์



รูปที่ 3.3 รายละเอียดระบบสะท้อนแสงอาทิตย์

3.4.1 ชุดเสารับน้ำหนัก

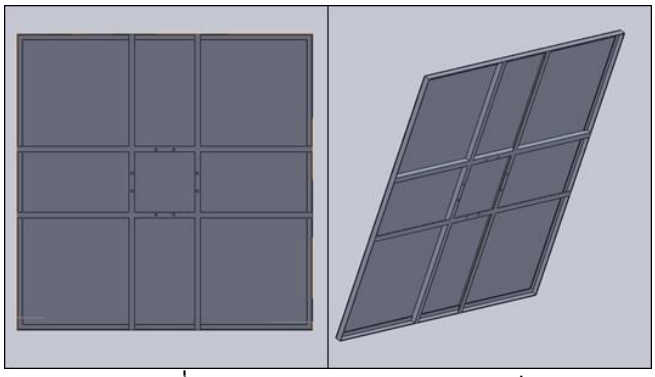
ใช้วัสดุท่อเหล็กกล้ามีตะเข็บ ขนาดเส้นผ่านศูนย์กลางภายนอก 0.082 เมตร หนา 0.003 เมตร สูง 0.85 เมตร ทำหน้าที่ในการรับน้ำหนักของชุดแผ่นสะท้อนแสงอาทิตย์และติดตั้งชุดมอเตอร์ในแกนหมุนที่ 1 ไว้ภายในปลายด้านบนของเสา ดังรูปที่ 3.4



รูปที่ 3.4 ชุดเสารับน้ำหนัก

3.4.2 แผ่นสะท้อนแสงอาทิตย์

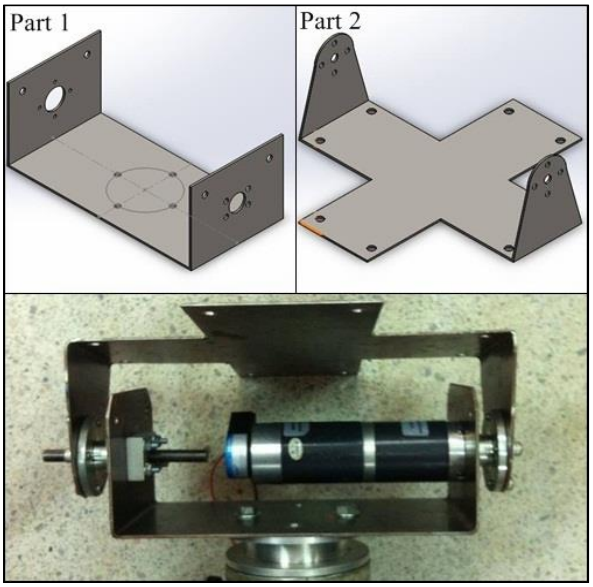
ประกอบไปด้วยโครงที่ทำมาจากอลูมิเนียมฉากและวัสดุสะท้อนแสง ซึ่งได้เลือใช้อะครีลิกฉาบปรอทสะท้อนแสง มีขนาดกว้าง 1 เมตร ยาว 1 เมตร หนา 2 มิลลิเมตร น้ำหนัก 2.5 กิโลกรัม วางยึดติดด้านบนของโครงอลูมิเนียม ดังรูปที่ 3.5



รูปที่ 3.5 แผ่นสะท้อนแสงอาทิตย์

3.4.3 ชุดกล่องจับยึดมอเตอร์และแผ่นสะท้อนแสงอาทิตย์

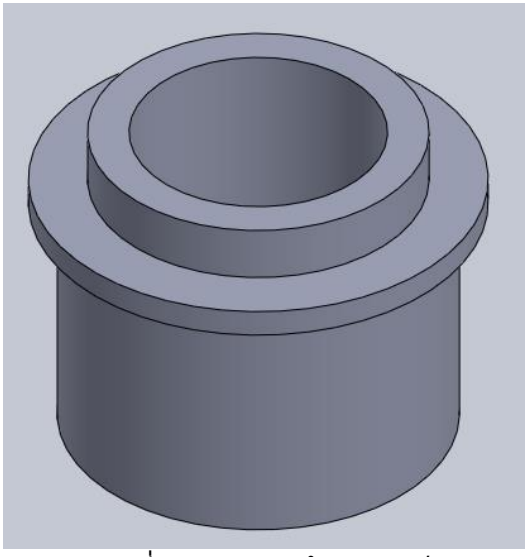
ทำจากวัสดุเหล็กกล้าแผ่น หนา 2 มิลลิเมตร ตัดพับขึ้นรูปดังรูปที่ 3.6 Part 1 มีขนาดกว้าง 0.1 เมตร ยาว 0.2 เมตร และสูง 0.08 เมตร ทำหน้าที่ติดตั้งและจับยึดมอเตอร์ในแกนหมุนที่ 2 เข้ากับตัวจับยึดแผ่นสะท้อนแสงอาทิตย์ด้านบนและชุดบรูชจับยึดมอเตอร์ของแกนหมุนที่ 1 ด้วย และ รูปที่ 3.6 Part 2 ทำจากวัสดุชนิดเดียวกัน มีขนาดกว้าง 0.25 เมตร ยาว 0.25 เมตร และสูง 0.087 เมตร ทำหน้าที่จับยึดและรับการส่งกำลังจากเพลาของมอเตอร์เพื่อควบคุมตำแหน่งของแผ่นสะท้อนแสงอาทิตย์



รูปที่ 3.6 ชุดกล่องจับยึดมอเตอร์และแผ่นสะท้อนแสงอาทิตย์

3.4.4 บรูชจับยึดมอเตอร์ด้านล่าง

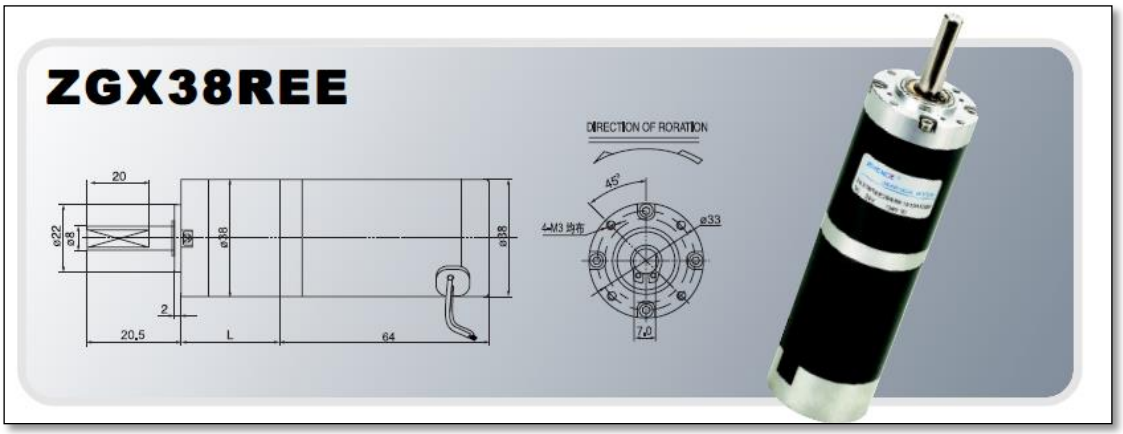
ทำจากอะลูมิเนียม มีขนาดเส้นผ่านศูนย์กลางภายนอก 82 มิลลิเมตร หนา 60 มิลลิเมตร ติดตั้งส่วนบนภายในเสา ทำหน้าที่ในการจับยึดมอเตอร์ในแกนหมุนที่ 1 เข้ากับเสาและชุดกล่องจับยึดมอเตอร์ฝนแกนหมุนที่ 2 ด้านบนเข้าด้วยกัน ดังรูปที่ 3.7



รูปที่ 3.7 บรูชจับยึดมอเตอร์

3.4.5 มอเตอร์

มอเตอร์ที่ใช้เป็นแบบดีซีเซอร์โวมอเตอร์ 24 โวลต์ ให้ความเร็วที่ 10 RPM ขนาดมอเตอร์ มีเส้นผ่านศูนย์กลาง 0.038 เมตร ยาว 0.142 เมตร ขนาดแกน 0.008 เมตร โดยมอเตอร์จะรับการจ่ายสัญญาณไฟฟ้าควบคุมการทำงานจากมอเตอร์ไดรว์ และมีเอ็นโคเดอร์ต่อกับมอเตอร์เพื่อใช้ในการวัดตำแหน่งการหมุนของมอเตอร์ และส่งสัญญาณกลับไปยังมอเตอร์ไดรว์อีกครั้ง ดังรูปที่ 3.8



รูปที่ 3.8 มอเตอร์ดีซีที่ใช้

3.5 อุปกรณ์ที่ใช้ในการควบคุม

3.5.1 บอร์ดไมโครคอนโทรลเลอร์ Arduino DUE

Arduino DUE เป็นบอร์ดไมโครคอนโทรลเลอร์รุ่นแรกสุดของ Arduino ซึ่งจะใช้ไมโครคอนโทรลเลอร์ 32 บิต ARM cortex M3 Arduino due มีพอร์ตอินพุต เอาท์พุตให้ใช้งานมากถึง 54 พอร์ต โดยรวมเอาความสามารถของโมดูล PWM ไว้ 12 ช่อง, อินพุตอะนาลอก 12 ช่อง, 4 UART, DAC 2 ชุด, I^2C 2 ชุด, โมดูล CAN และรองรับการทำงานกับอุปกรณ์ USB ในแบบ USB-OTG ซึ่งเป็นบอร์ดที่มีความสามารถในการทำงานสูง และ ต้องใช้กับซอฟต์แวร์ Arduino IDE เวอร์ชั่น 1.5 ขึ้นไป ซึ่งคุณสมบัติของบอร์ดไมโครคอนโทรลเลอร์รุ่น AT91SAM3X8E มีรายละเอียดดังนี้

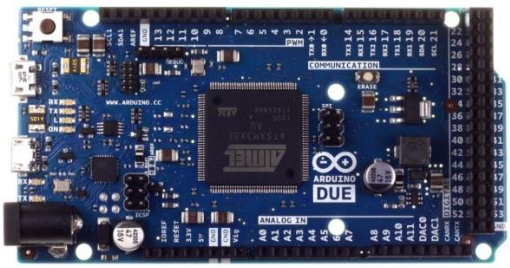
1) ไมโครคอนโทรลเลอร์ AT91SAM3X8E จาก Atmel หน่วยความจำแฟลช 512 กิโลไบต์, แรม 96 กิโลไบต์ มีความสามารถ USB-OTG ทำหน้าที่เป็นอุปกรณ์ USB โฮสต์ เพื่อต่อกับเมาส์ คีย์บอร์ด และสมาร์ตโฟนได้

2) ความถี่สัญญาณนาฬิกา 84MHz ไฟเลี้ยงทำงาน 3.3 V และ ย่านไฟเลี้ยงอินพุต 7 ถึง 12 V

3) พอร์ตอินพุตเอาต์พุต 54 ช่อง (รวมเอาต์พุต PWM 12 ช่อง)

4) อินพุตอะนาลอก 12 ช่อง ความละเอียด 12 บิต รับแรงดัน 0 ถึง +3.3V และเอาต์พุตอะนาลอก (DAC) 2 ช่อง ความละเอียด 12 บิต ให้แรงดัน 0 ถึง +3.3V

5) บัสสื่อสารข้อมูล I^2C 2 ชุด, SPI, JTAG, CAN, UART 4 ชุด และติดต่อคอมพิวเตอร์ พอร์ต USB โดยใช้สาย microUSB



รูปที่ 3.9 บอร์ดไมโครคอนโทรลเลอร์ Arduino DUE [13]

3.5.2 มอเตอร์ไดร์ฟ

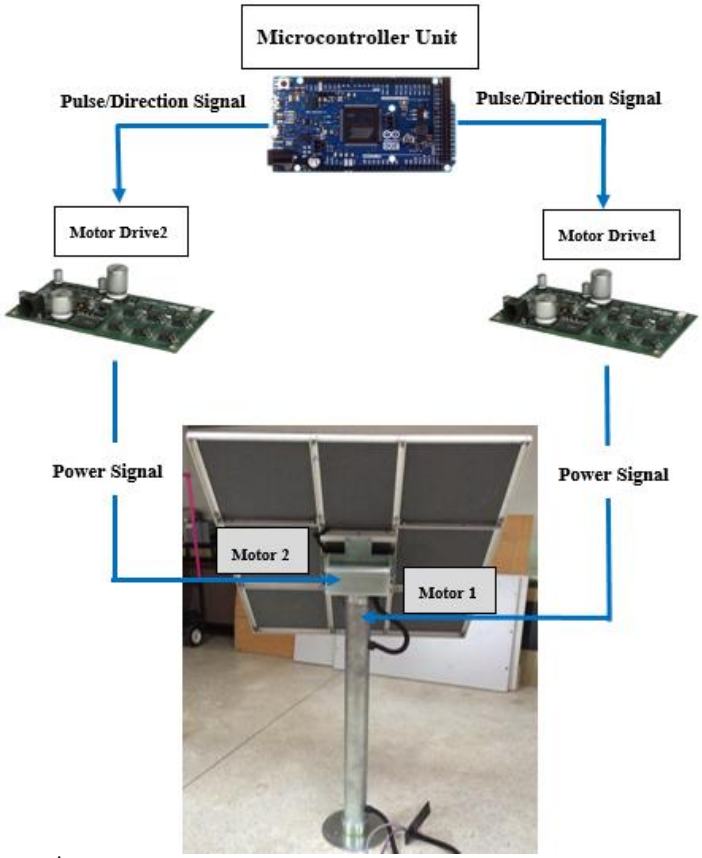
มอเตอร์ไดร์ฟที่ ใช้ในการควบคุมการทำงานของมอเตอร์ เป็นของบริษัท Copley Controls รุ่น ASC-055-18 ใช้แรงดันไฟ 20-55 โวตท์ สามารถควบคุมตำแหน่ง ความเร็วและแรงบิดได้ และมอเตอร์ไดร์ฟจะควบคุมการจ่ายสัญญาณไฟฟ้าเข้าสู่มอเตอร์ พร้อมทั้งรับสัญญาณเอ็นโคดเดอร์ที่ต่อกับตัวมอเตอร์และสัญญาณดิจิตอลจากลิมิตสวิทช์เพื่อใช้ในการควบคุมมอเตอร์ สำหรับโครงงานวิจัยนี้ได้กำหนดให้มอเตอร์ไดร์ฟทำงานควบคุมตำแหน่งการหมุนของมอเตอร์ โดยจะรับสัญญาณคำสั่งการหมุนมาจากไมโครคอนโทรลเลอร์ รูปแบบสัญญาณที่ส่งมาจาก

ไมโครคอนโทรลเลอร์จะเป็นสัญญาณพลัส โดยถ้าไมโครคอนโทรลเลอร์ส่งมา 10 พลัส มอเตอร์จะหมุน 10/4 พลัสของสัญญาณเอ็นโคเดอร์

รูปที่ 3.10 มอเตอร์ไดร์ฟ [14]

3.6 ระบบสัญญาณการควบคุมตัวสะท้อนแสงอาทิตย์

ในการควบคุมการทำงานของระบบสะท้อนแสงอาทิตย์ได้ใช้ไมโครคอนโทรลเลอร์ Arduino รุ่น DUE เป็นตัวประมวลผลการทำงานหลัก และใช้บอร์ด Shield ในการสร้างสัญญาณนาฬิกาจริงร่วมกัน โดยที่บอร์ด Arduino จะรับค่าเวลา วัน เดือน ปี จากบอร์ด Shield เพื่อใช้ในการคำนวณหาตำแหน่งดวงอาทิตย์ (มุมอะซิมุธและมุมอัลติจูด) และคำนวณหามุมหมุนของระบบสะท้อนแสงอาทิตย์ เมื่อได้ค่ามุมในการหมุน บอร์ด Arduino จะสร้างสัญญาณพัลส์ และทิศทาง ส่งไปยังบอร์ดขับมอเตอร์ ซึ่งได้ใช้บอร์ดยี่ห้อ Copley Controls รุ่น ASC-055-18 เพื่อควบคุมให้ดีซีเซอร์โวมอเตอร์ที่ถูกติดตั้งภายในตัวระบบสะท้อนแสงอาทิตย์หมุนเคลื่อนที่ให้สะท้อนแสงไปยังตำแหน่งที่ต้องการ



รูปที่ 3.11 ระบบสัญญาณการควบคุมตัวสะท้อนแสงอาทิตย์

บทที่ 4
การทดลองและผลลัพธ์

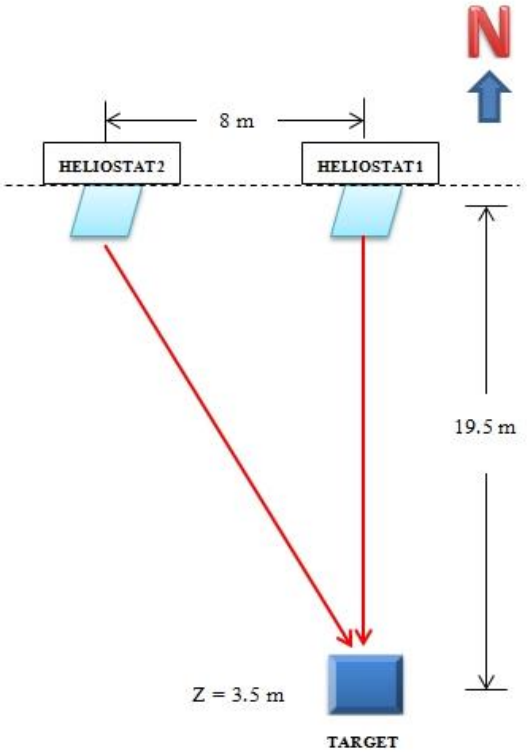
ในบทนี้เป็นการทดลองการทำงานของระบบควบคุมสำหรับเสาสะท้อนแสงอาทิตย์ ซึ่งในการทดลองนี้ได้แบ่งออกเป็น 4 การทดลอง คือ 1.การทดสอบหาระยะเวลาที่เหมาะสมในการหมุนของระบบสะท้อนแสงอาทิตย์ 2. การทดสอบระบบสะท้อนแสงอาทิตย์ 1 ตัว ตั้งสะท้อนในแนวตรงกับเป้าหมาย 3. การทดสอบระบบสะท้อนแสงอาทิตย์ 1 ตัว ตั้งสะท้อนในแนวเฉียงกับเป้าหมาย และ 4. การทดสอบระบบสะท้อนแสงอาทิตย์ทั้ง 2 แบบพร้อมกัน

4.1 การติดตั้งระบบสะท้อนแสงอาทิตย์

สถานที่ติดตั้งและทดสอบระบบสะท้อนแสงอาทิตย์นั้นได้ใช้สนามบริเวณหน้าภาควิชาวิศวกรรมเครื่องกล มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรีเป็นจุดติดตั้งระบบสะท้อนแสงอาทิตย์ทั้ง 2 ตัว

4.1.1 การติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 1 จะติดตั้งในแนวตรงตามทิศเหนือ-ใต้ กับเป้าหมายซึ่งตำแหน่งติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 1 อยู่ที่ ละติจูด 14.036711 องศาเหนือ, ลองติจูด 100.724439 องศาตะวันออก โดยกำหนดให้มีเป้าหมายของการสะท้อนอยู่ที่ $V_T = -19.5i + 0j + 3.5k$ เมตร

4.1.2 การติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 2 นั้นจะอยู่ในแนวเดียวกับระบบสะท้อนแสงอาทิตย์ตัวที่ 1 แต่จะเฉียงกับเป้าหมายเป็นระยะทาง 8 เมตร ซึ่งตำแหน่งติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 2 อยู่ที่ ละติจูด 14.036711 องศาเหนือ, ลองติจูด 100.724362 องศาตะวันออก โดยกำหนดให้มีเป้าหมายของการสะท้อนอยู่ที่ $V_T = -19.5i + 8j + 3.5k$ เมตร



รูปที่ 4.1 ผังการติดตั้งระบบสะท้อนแสงอาทิตย์

รูปที่ 4.2 การติดตั้งระบบสะท้อนแสงอาทิตย์

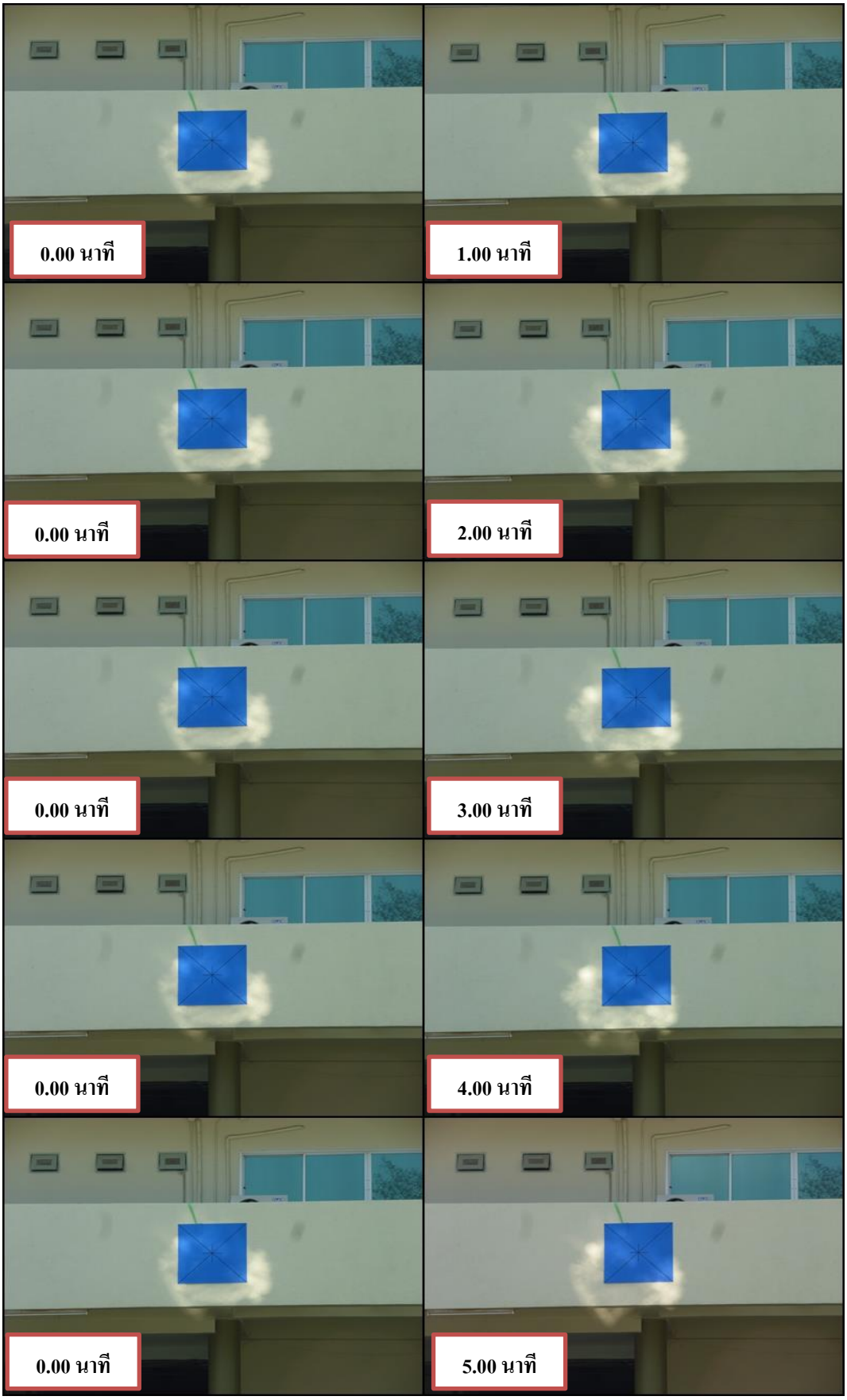
รูปที่ 4.3 ตำแหน่งเป้าหมายของการทดสอบ

4.2 การทดสอบหาเวลาที่เหมาะสมในการทำงานของระบบสะท้อนแสงอาทิตย์

ในการทดสอบนี้นั้นจะเป็นการทดสอบระบบสะท้อนแสงอาทิตย์เพื่อหาระยะเวลาที่เหมาะสมในการหมุนของแผงสะท้อน ว่าควรจะใช้เวลาในการหยุดหมุนของมอเตอร์เท่าใด ที่จะไม่ส่งผลต่อความผิดพลาดของแสงที่สะท้อนไปยังเป้าหมาย เพื่อที่จะได้นำเวลาที่เหมาะสมไปทำการเขียนโปรแกรมควบคุมการทำงานจริงของระบบสะท้อนแสงอาทิตย์ต่อไป ทำการทดสอบในวันที่ 30 มกราคม 2557

4.1.1 ขั้นตอนการทดสอบหาระยะเวลาที่เหมาะสมในการหยุดหมุนของระบบสะท้อนแสงอาทิตย์

1) ทำการติดตั้งระบบสะท้อนแสงอาทิตย์โดยวางในแนวตรงกับเป้าหมาย
2) เขียนโปรแกรมและให้ระบบสะท้อนแสงอาทิตย์ทำงานและสะท้อนแสงไปยังเป้าหมาย โดยการให้ระยะเวลาในการหยุดหมุนของมอเตอร์ของระบบสะท้อนแสงอาทิตย์เป็น 5 นาที
3) บันทึกภาพของแสงที่สะท้อนไปยังเป้าหมาย ทุกๆ 1 นาที
4) เปรียบเทียบผลที่ได้จากการทดสอบ



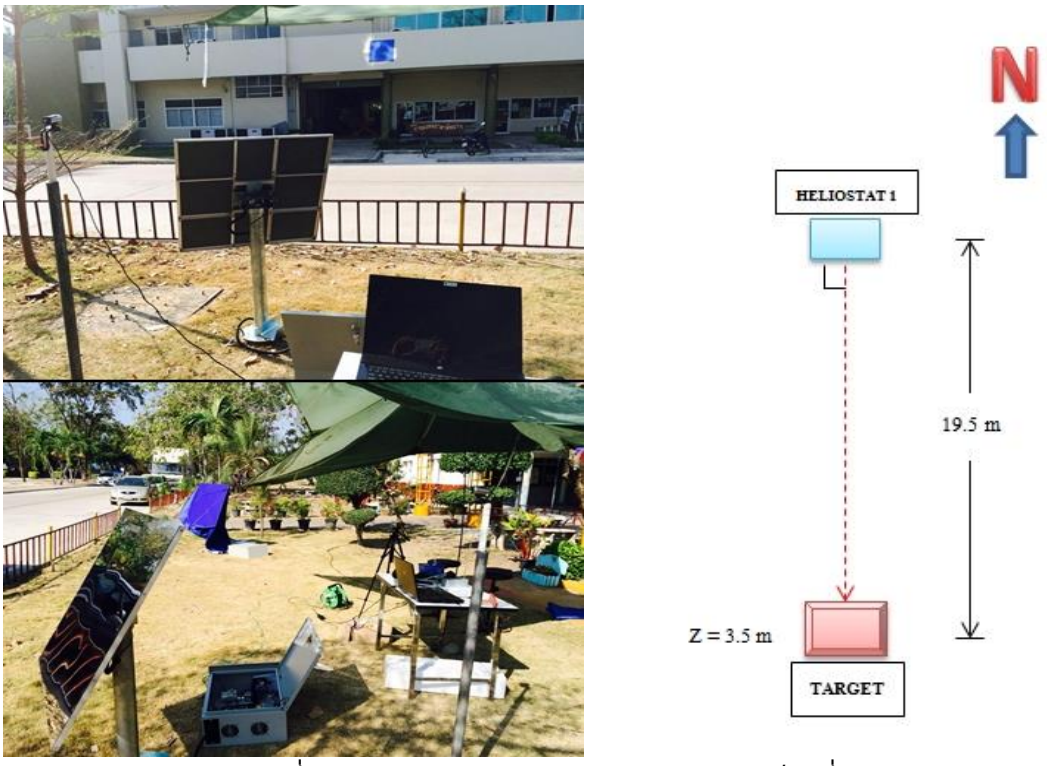
รูปที่ 4.4 ผลการทดสอบหาเวลาที่เหมาะสมในการทำงานของระบบสะท้อนแสงอาทิตย์

4.1.2 ผลที่ได้จากการทดสอบหาเวลาที่ใช้ในการทำงานของระบบสะท้อนแสงอาทิตย์

จากผลที่ได้จากการทดสอบจะเห็นได้ว่าช่วงระยะเวลาในการหยุดทำงานของมอเตอร์ที่เหมาะสมที่จะไม่ส่งผลต่อความคลาดเคลื่อนจากการสะท้อนแสงนั้นจะอยู่ที่ประมาณ 2 นาที ซึ่งหากใช้เวลานานเกินกว่านี้จะทำให้แสงที่สะท้อนไปนั้นเกิดความคลาดเคลื่อนจากเป้าหมายมากขึ้น แต่หากใช้เวลาน้อยกว่า 2 นาทีนั้น ผลจากการลดเวลาก็จะส่งผลต่อแสงสะท้อนที่ทำให้เกิดความเคลื่อนจากเป้าหมายน้อยมาก และเพื่อเป็นการลดจำนวนครั้งในการทำงานของมอเตอร์ เพื่อเป็นการประหยัดพลังงานไฟฟ้าแล้ว ดังนั้นจึงเลือกใช้ระยะเวลาในการหยุดหมุนของมอเตอร์ (Delay) เท่ากับ 2 นาที ต่อการทำงานของมอเตอร์ 1 ครั้ง

4.3 การทดสอบการสะท้อนแสงของระบบสะท้อนแสงอาทิตย์ตัวที่ 1

ในการทดสอบการสะท้อนแสงระบบสะท้อนแสงอาทิตย์ตัวที่ 1 นั้น จะทำการติดตั้งระบบสะท้อนแสงอาทิตย์ในแนวตรงตามทิศเหนือ-ใต้ กับเป้าหมายซึ่งตำแหน่งติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 1 อยู่ที่ ละติจูด 14.036711 องศาเหนือ, ลองติจูด 100.724439 องศาตะวันออก โดยมีเป้าหมายของการสะท้อนอยู่ที่ $V_T = -19.5i + 0j + 3.5k$ เมตร ทำการทดสอบในวันที่ 31 มกราคม 2557



รูปที่ 4.5 การทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1

4.3.1 ขั้นตอนการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1

1) ติดตั้งระบบสะท้อนแสงอาทิตย์ในตำแหน่งที่กำหนดไว้

2) เขียนโปรแกรมควบคุมให้กับระบบสะท้อนแสงอาทิตย์โดยให้มอเตอร์ทั้ง 2 แกนของระบบทำงานในเวลาทุกๆ 2 นาที

3) เริ่มทำการทดสอบระบบสะท้อนแสงอาทิตย์ตั้งแต่เวลา 9.00 น. ถึง 16.00 น.

4) เก็บผลจากการทดสอบโดยการบันทึกภาพของแสงที่สะท้อนไปยังเป้าหมายทุกๆ 30 นาทีและบันทึกวิดีโอตลอดการทดสอบ

รูปที่ 4.6 ระนาบของระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และตำแหน่งเป้าหมาย

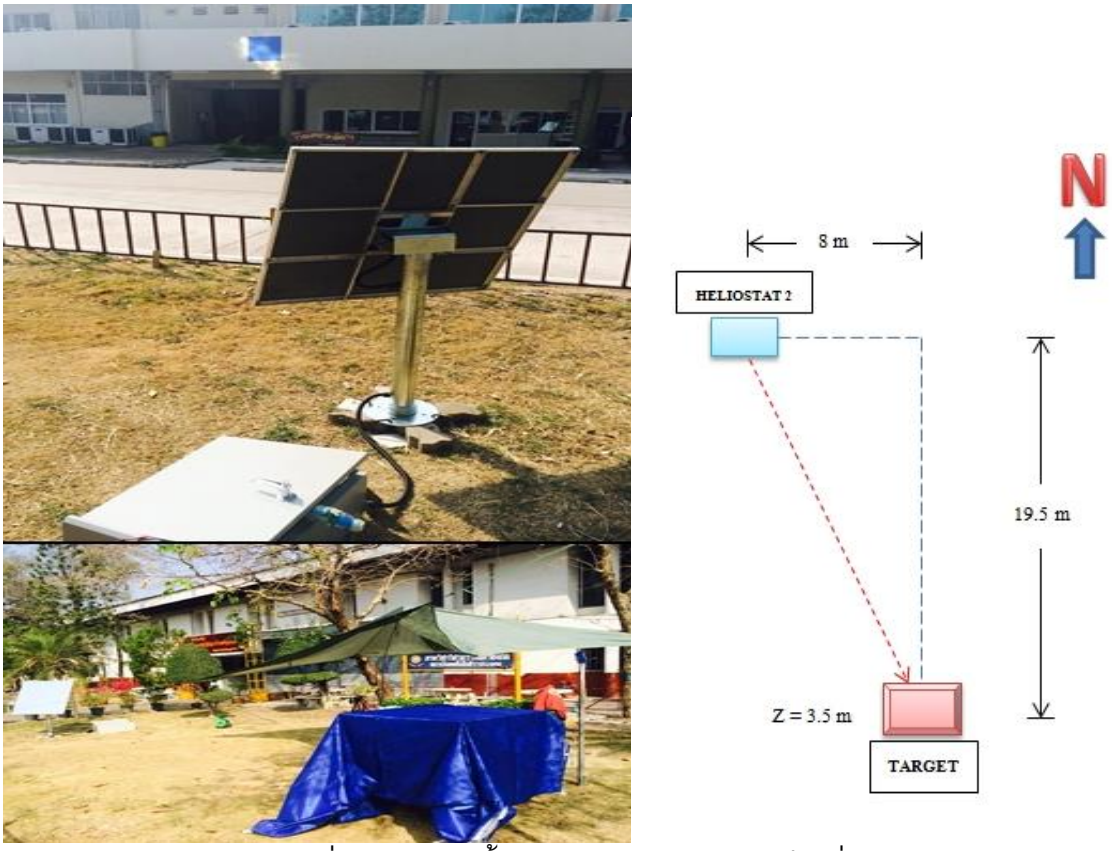
4.3.2 ผลที่ได้จากการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1

จากผลการทดสอบการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 1 ซึ่งตั้งสะท้อนในแนวตรงกับเป้าหมาย จะเห็นได้ว่าระบบสะท้อนแสงอาทิตย์สามารถทำงานและสะท้อนแสงไปยังเป้าหมายที่ต้องการได้โดยตำแหน่งที่แสงสะท้อนไปนั้นจะอยู่ตำแหน่งเดิมตลอดทั้งวัน แต่อาจจะมีความคลาดเคลื่อนจากเป้าหมายเล็กน้อย ทำให้ต้องปรับตั้งตัวระบบสะท้อนแสงอาทิตย์ใหม่เพื่อให้แสงสะท้อนตรงกับเป้าหมาย

รูปที่ 4.7 ผลการทดสอบการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 1

4.4 การทดสอบการสะท้อนแสงของระบบสะท้อนแสงอาทิตย์ตัวที่ 2

ในการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 2 นั้น จะทำการติดตั้งระบบสะท้อนแสงอาทิตย์ในแนวเดียวกันกับระบบสะท้อนแสงอาทิตย์ตัวที่ 1 แต่จะเฉียงกับเป้าหมายเป็นระยะทาง 8 เมตร ซึ่งตำแหน่งติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 2 อยู่ที่ ละติจูด 14.036711 องศาเหนือ, ลองติจูด 100.724362 องศาตะวันออก โดยมีเป้าหมายของการสะท้อนอยู่ที่ $V_T = -19.5i + 8j + 3.5k$ เมตร ทำการทดสอบในวันที่ 1 กุมภาพันธ์ 2557



รูปที่ 4.8 การติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 2

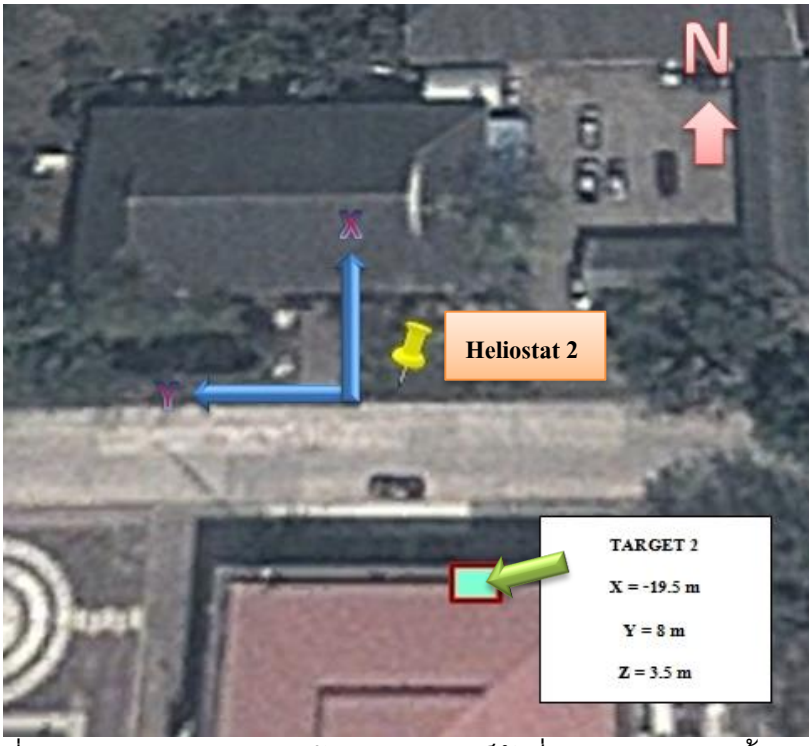
4.4.1 ขั้นตอนการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 2

1) ติดตั้งระบบสะท้อนแสงอาทิตย์ในตำแหน่งที่กำหนดไว้

2) เขียนโปรแกรมควบคุมให้กับระบบสะท้อนแสงอาทิตย์โดยให้มอเตอร์ทั้ง 2 แกนของระบบทำงานในเวลาทุกๆ 2 นาที

3) เริ่มทำการทดสอบระบบสะท้อนแสงอาทิตย์ตั้งแต่เวลา 9.00 น. ถึง 16.00 น.

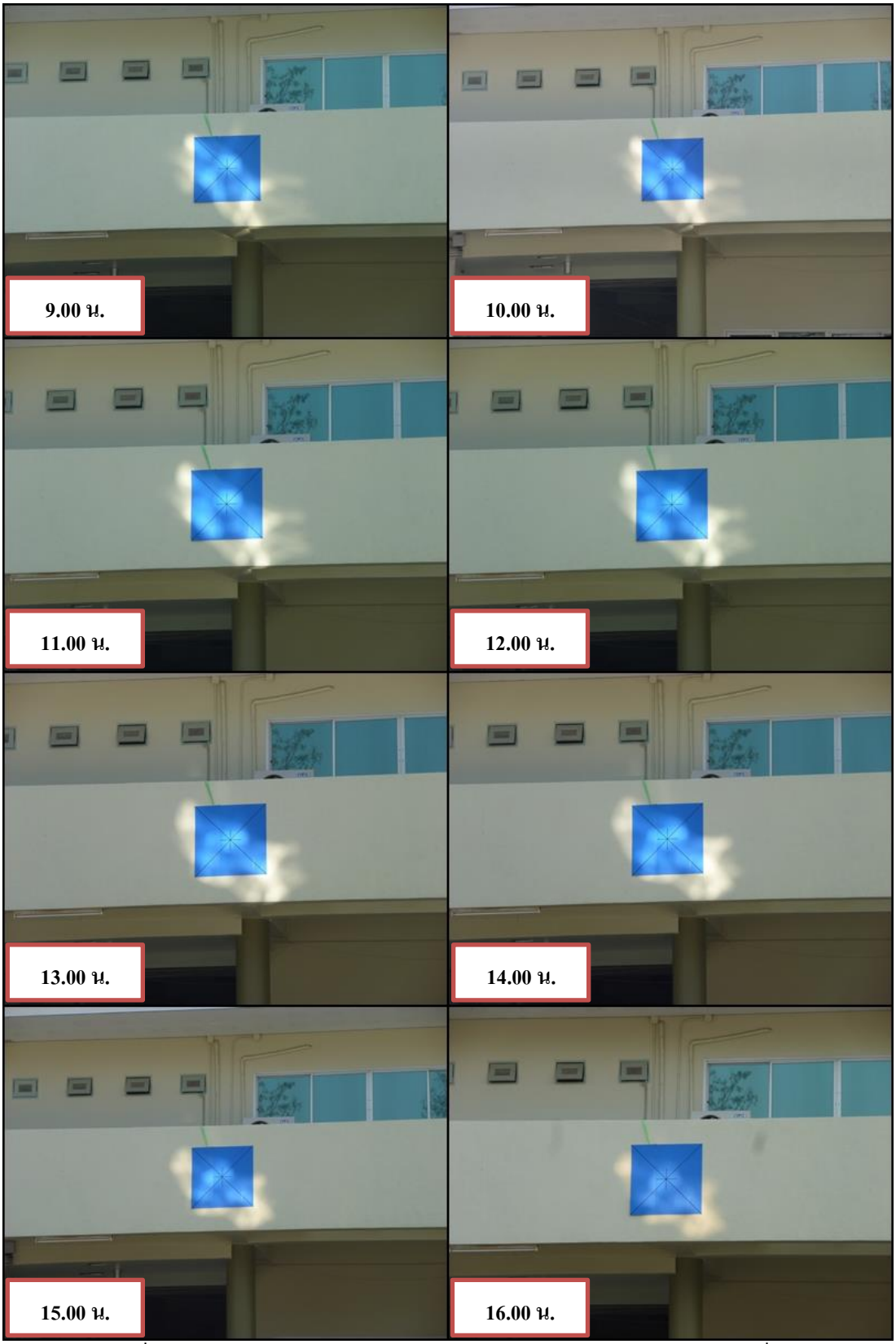
4) เก็บผลจากการทดสอบโดยการบันทึกภาพของแสงที่สะท้อนไปยังเป้าหมายทุกๆ 30 นาทีและบันทึกวิดีโอตลอดการทดสอบ



รูปที่ 4.9 ระนาบของระบบสะท้อนแสงอาทิตย์ตัวที่ 2 และตำแหน่งเป้าหมาย

4.4.2 ผลที่ได้จากการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 2

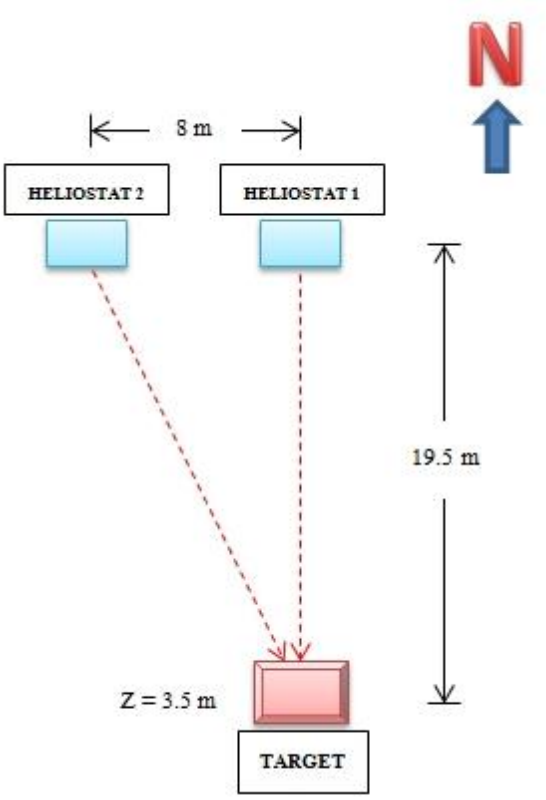
จากผลการทดสอบการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 2 ซึ่งตั้งสะท้อนในแนวเฉียงกับเป้าหมาย จะเห็นได้ว่าระบบสะท้อนแสงอาทิตย์สามารถทำงานและสะท้อนแสงไปยังเป้าหมายที่ต้องการได้โดยตำแหน่งที่แสงสะท้อนไปนั้นจะอยู่ตำแหน่งเดิมตลอดทั้งวัน แต่อาจจะมีความคลาดเคลื่อนจากเป้าหมายเล็กน้อย ซึ่งต้องปรับตั้งตัวระบบสะท้อนแสงอาทิตย์ใหม่เพื่อให้แสงสะท้อนตรงกับเป้าหมาย



รูปที่ 4.10 ผลการทดสอบการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 2

4.5 การทดสอบการสะท้อนแสงของระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และ 2

ในการทดสอบระบบสะท้อนแสงอาทิตย์ทั้ง 2 ตัวพร้อมกันนั้น จะทำการติดตั้งในตำแหน่งเดิมที่กำหนดไว้ แต่ในกรณีนี้จะให้ระบบสะท้อนแสงอาทิตย์ทั้ง 2 ตัวทำงานพร้อมกันโดยสะท้อนแสงไปยังเป้าหมายเดียวกัน ซึ่งมีเป้าหมายของการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 1 อยู่ที่ $V_T = -19.5i + 0j + 3.5k$ เมตร และเป้าหมายของการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 2 อยู่ที่ $V_T = -19.5i + 8j + 3.5k$ เมตร ทำการทดสอบในวันที่ 2 กุมภาพันธ์ 2557



รูปที่ 4.11 การติดตั้งระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และ 2

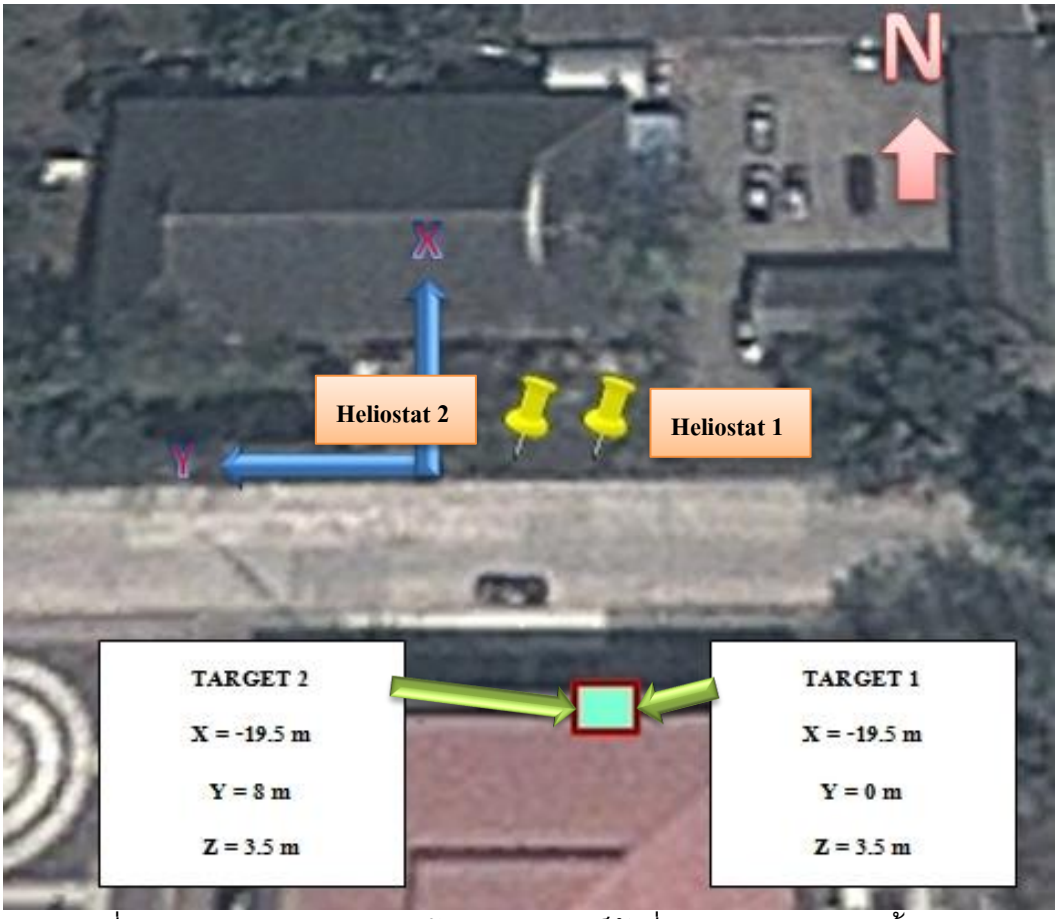
4.5.1 ขั้นตอนการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และ 2 พร้อมกัน

1) ติดตั้งระบบสะท้อนแสงอาทิตย์ทั้ง 2 ตัวในตำแหน่งที่กำหนดไว้

2) เขียนโปรแกรมควบคุมให้กับระบบสะท้อนแสงอาทิตย์ทั้ง 2 ตัวโดยให้มอเตอร์ทั้ง 2 แกนของระบบทำงานในเวลาทุกๆ 2 นาที

3) เริ่มทำการทดสอบระบบสะท้อนแสงอาทิตย์ตั้งแต่เวลา 9.00 น. ถึง 16.00 น.

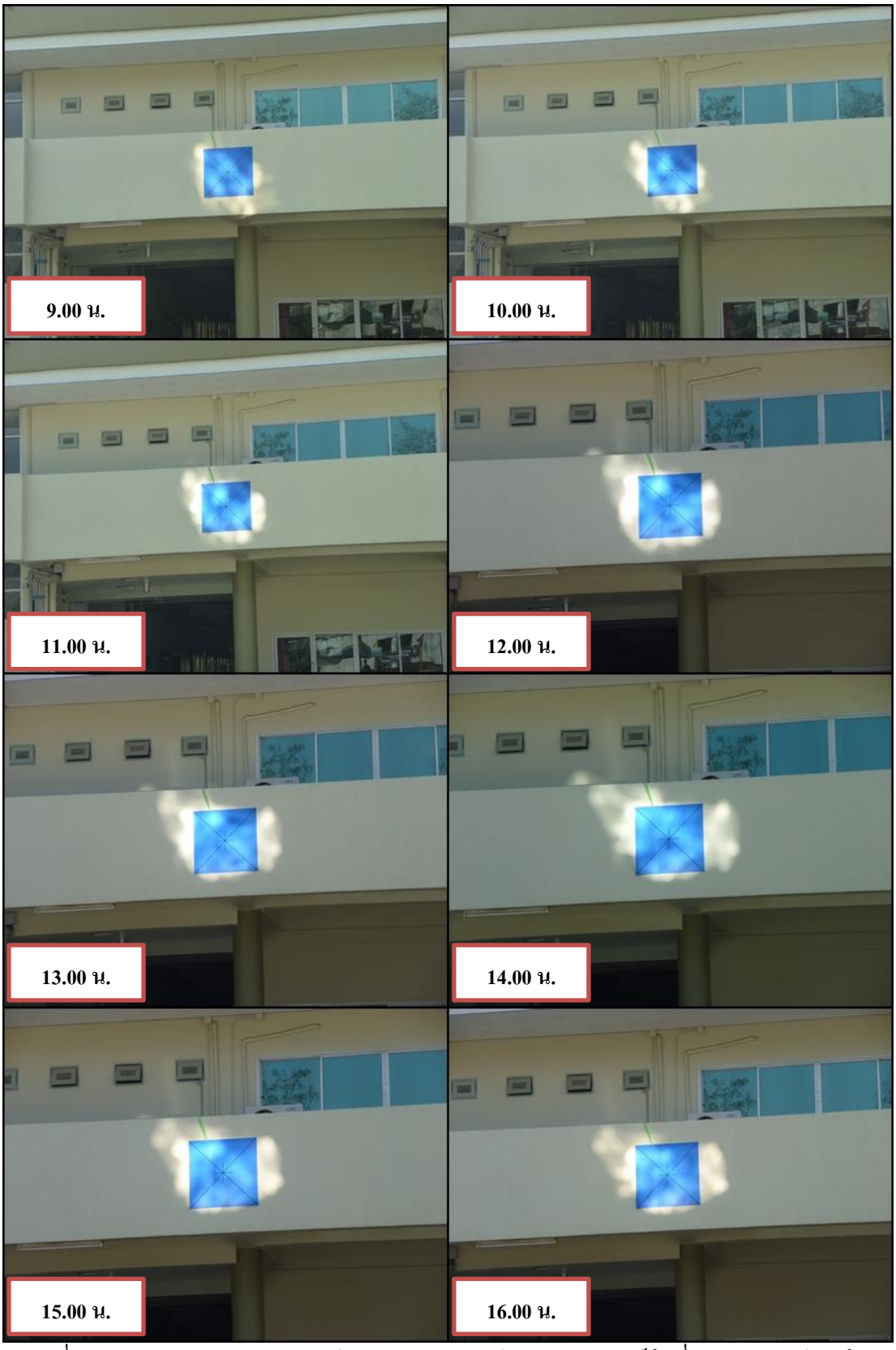
4) เก็บผลจากการทดสอบโดยการบันทึกภาพของแสงที่สะท้อนไปยังเป้าหมายทุกๆ 30 นาทีและบันทึกวิดีโอตลอดการทดสอบ



รูปที่ 4.12 ระนาบของระบบสะท้อนแสงอาทิตย์ตัวที่ 1, 2 และตำแหน่งเป้าหมาย

4.5.2 ผลที่ได้จากการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และ 2 พร้อมกัน

จากผลการทดสอบการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และ2 พร้อมกัน โดยตัวที่ 1 ตั้งสะท้อนในแนวตรงกับเป้าหมาย และตัวที่ 2 ตั้งสะท้อนในแนวเฉียงกับเป้าหมาย ซึ่งทั้ง 2 ตัวนั้นสะท้อนไปยังเป้าหมายเดียวกัน ซึ่งจากรูปผลการทดสอบจะเห็นได้ว่าระบบสะท้อนแสงอาทิตย์ ทั้ง 2 ตัว สามารถทำงานและสะท้อนแสงไปยังเป้าหมายที่ต้องการได้โดยตำแหน่งที่แสงสะท้อนไปนั้นจะอยู่ตำแหน่งเดิมตลอดทั้งวัน แต่อาจจะมีความคลาดเคลื่อนจากเป้าหมายเล็กน้อย ซึ่งต้องปรับตั้งตัวระบบสะท้อนแสงอาทิตย์ใหม่เพื่อให้แสงสะท้อนตรงกับเป้าหมาย



รูปที่ 4.13 ผลการทดสอบการสะท้อนของระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และ 2 พร้อมกัน

บทที่ 5
บทสรุป

5.1 สรุปผล

โครงงานวิจัยนี้เป็นการออกแบบสร้างระบบสะท้อนแสงอาทิตย์และเขียนโปรแกรมควบคุมการทำงานของมอเตอร์ทั้ง 2 แกน โดยใช้การคำนวณจากสมการทางคณิตศาสตร์เพื่อหาตำแหน่งมุมของดวงอาทิตย์ ทั้งมุมอัลติจูด และมุมอะซิมุธ จากนั้นแปลงค่ามุมทั้ง 2 นี้เป็นมุมในการหมุนของเสาสะท้อนแสงอาทิตย์ โดยอ้างอิงจากเวกเตอร์ของเป้าหมาย จากนั้นแปลงค่ามุมในการหมุนของระบบสะท้อนแสงอาทิตย์มาเป็นมุมในการหมุนของมอเตอร์ทั้ง 2 แกน

ในการเขียนโปรแกรมคำนวณหาตำแหน่งของดวงอาทิตย์ และตำแหน่งในการหมุนของระบบสะท้อนแสงอาทิตย์ รวมทั้งสั่งควบคุมการทำงานของมอเตอร์นั้น จะใช้บอร์ดไมโครคอนโทรลเลอร์ Arduino DUE เป็นตัวประมวลผล จากนั้นแปลงค่ามุมในการหมุนของมอเตอร์มาเป็นสัญญาณพัลส์ส่งไปยังมอเตอร์ไดร์ฟ เพื่อควบคุมตำแหน่งในการหมุนมอเตอร์ทั้ง 2 ตัวของระบบสะท้อนแสงอาทิตย์ให้ได้ตำแหน่งตรงตามที่ไมโครคอนโทรลเลอร์คำนวณและสั่งออกมา

การทดลองแบ่งออกเป็น 4 ส่วนคือส่วนแรกเป็นการทดสอบหาระยะเวลาที่เหมาะสมในการทำงานของระบบสะท้อนแสงอาทิตย์ว่าควรจะให้ระบบสะท้อนแสงอาทิตย์ทำงานและสะท้อนแสงอาทิตย์ในระยะเวลาทุกๆ กี่นาที และส่วนที่ 2 คือ การทดสอบการสะท้อนแสงไปยังเป้าหมายของระบบสะท้อนแสงอาทิตย์ ซึ่งแบ่งการทดสอบออกเป็น 3 ส่วนคือ การทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1 ตั้งสะท้อนในแนวตรงกับเป้าหมาย, การทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 2

ตั้งสะท้อนในแนวเฉียงกับเป้าหมาย และสุดท้ายเป็นการทดสอบระบบสะท้อนแสงอาทิตย์ตัวที่ 1 และตัวที่ 2 ให้ทำงานพร้อมกันโดยสะท้อนแสงไปยังเป้าหมายเดียวกัน

จากการทดสอบหาระยะเวลาที่เหมาะสมในการทำงานของระบบสะท้อนแสงอาทิตย์ในหัวข้อ 4.2 ซึ่งจากผลการทดสอบจะเห็นได้ว่าช่วงระยะเวลาในการหยุดทำงานของมอเตอร์ที่เหมาะสมที่จะไม่ส่งผลต่อความคลาดเคลื่อนของการสะท้อนแสงมากเกินไปนั้นจะอยู่ที่เวลาประมาณ 2 นาที ซึ่งหากใช้เวลานานเกินกว่านี้จะทำให้แสงที่สะท้อนไปยังเป้าหมายนั้นเกิดความคลาดเคลื่อนจากเป้าหมายมาก แต่หากใช้เวลาน้อยกว่า 2 นาทีนั้น ผลจากการลดเวลาของการหยุดทำงานจะส่งผลต่อแสงที่สะท้อนไปยังเป้าหมายและทำให้เกิดความเคลื่อนจากเป้าหมายเล็กน้อย และเพื่อเป็นการลดจำนวนครั้งในการทำงานของมอเตอร์แล้ว ดังนั้นจึงควรเลือกใช้ระยะเวลาในการหยุดหมุนของมอเตอร์ (Delay) เท่ากับ 2 นาที ต่อการทำงานของมอเตอร์ 1 ครั้ง

การทดสอบการสะท้อนแสงของระบบสะท้อนแสงอาทิตย์ทั้ง 2 ตัวในหัวข้อ 4.3, 4.4 และ 4.5 นั้น จากผลการทดสอบจะเห็นได้ว่าการสะท้อนแสงอาทิตย์ไปยังเป้าหมายของระบบสะท้อนแสงอาทิตย์จากการทดสอบทั้ง 3 แบบ ระบบสามารถสะท้อนแสงไปยังเป้าหมายที่ต้องการได้ ทั้งในตำแหน่งตั้งตรงกับเป้าหมาย และตำแหน่งตั้งเฉียงกับเป้าหมายด้วย และแสงที่สะท้อนมีความคลาดเคลื่อนจากเป้าหมายเพียงเล็กน้อย ซึ่งส่วนนี้สามารถปรับตั้งที่ตัวระบบสะท้อนแสงอาทิตย์เพื่อให้แสงที่สะท้อนตรงเป้าหมายได้ จากผลการทดสอบนี้ทำให้เห็นได้ว่าสามารถนำไมโครคอนโทรลเลอร์มาประยุกต์ใช้กับระบบสะท้อนแสงอาทิตย์โดยการคำนวณหาตำแหน่งในการหมุนของมอเตอร์จากสมการทางคณิตศาสตร์ได้ พร้อมทั้งสามารถควบคุมการทำงานและสะท้อนแสงอาทิตย์ไปยังเป้าหมายได้ในทุกสภาพอากาศ โดยระบบจะหมุนปรับตามดวงอาทิตย์เพื่อสะท้อนแสงไปยังเป้าหมายที่ต้องการในเวลาทุกๆ 2 นาที

5.2 ข้อเสนอแนะและแนวทางการพัฒนา

5.2.1 ควรใช้มอเตอร์ที่มีหัวเกียร์ที่มีค่าความคลอนต่ำ เพื่อลดความคลาดเคลื่อนในการสะท้อนแสงไปยังเป้าหมายด้วย

5.2.2 ควรมีการพัฒนาโปรแกรมเพื่อให้ระบบสามารถทำงานได้ตลอด 24 ชั่วโมง

บรรณานุกรม

[1] Grena, R., “An Algorithm for the Computation of the Solar Position”, *Solar Energy*, vol. 82, 2008.

[2] Reda, I. & Andreas, A., *Solar Position Algorithm for Solar Radiation Applications*, Technical Report of National Renewable Energy Laboratory, 2008.

[3] Zang, C., Wang, Z. & Liu, X., “Design and analysis of a novel heliostat structure”, Int. Conf. on Sustainable Power Generation and Supply, Nanjing, China, 6-7 April 2009.

[4] Chaib, A., Kesraoui, M. & Kechadi, E. “Heliostat Orientation System using a PLC based Robot Manipulator”, *Int. Conf. and Exh. On Ecological Vehicles and Renewable Energies (EVER)*, Monte Cario, 27-30 March 2013.

[5] Rubio, F.R., Ortega, M. G. & Lopez-Martinez, M., Application of New Control Strategy for Sun Tracking”, *Solar Energy*, vol. 48, 2007.

[6] Enrile, J., Ceron, F., Valera, P. & Osuna, R., “Prothelios: Heliostat for Large PV Plants”, *Proc. 3rd Conf. on Photovoltaic Energy Conversion*, Osaga, Japan, May 2003.

[7] Baheti, R.S. and Scott, P.F., “Design of self – calibrating Controllers for Heliostats in a Solar Power Plant,” *IEEE Transactions on Automatic Control*, Vol. ac – 25, pp. 1091 – 1097, 1980.

[8] Janthong, M. & Praditapai, A., “Control System of the Sun Reflection for Central Receiver System, *KKU Res. J.*, vol 17(3), 2012, pp. 432-442.

[9] โครงการ การเรียนรู้เรื่องวิทยาศาสตร์โลกและอวกาศ, **การหมุนเวียนของบรรยากาศ และอิทธิพลของฤดูกาล** (Online), Available: http:/203.172.208.242/tatalad/subject/Science/Earth%20Science/atmosphere/atm_circulation/atm _circulation/atm_circulation.htm (2 กุมภาพันธ์ 2557)

[10] ดาราศาสตร์และอวกาศสำหรับคนไทย, **ความสัมพันธ์ระหว่างโลกและดวงอาทิตย์** (Online), Available: http:// webstory.netfirms.com/story/solar1sunandearth.ht (2 กุมภาพันธ์ 2557)

[11] ภาควิชิฟิสิกส์ คณะวิทยาศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี, **มุมอซิมุธ** (Online), Available: http:// www.neutron.rmutphysics.com (2 กุมภาพันธ์ 2557).

[12] กรมพัฒนาพลังงานทดแทนและอนุรักษ์พลังงาน กระทรวงพลังงาน, **เทคโนโลยีพลังงานแสงอาทิตย์**.

[13] Website: http://arduino.cc/en/Main/ArduinoBoardDue#.UxXcfPmSzB1, 10/01/2014.

[14] Manual: Accelus™ Card, Copley Controls Corp.

[16] Website: http://www.primusthai.com/pdf_artical/1212032165_primust.pdf, 05/05/2013.

[17] Website: http://www.harvyrack.com/th/selective.php, 10/05/2013.

[18] Website: http://www.cisco-eagle.com/catalog/, 10/05/2013

[19] Website: http://www.matrixok.com/products/racks/pushback/, 10/05/2013.

[20] Website: http://www.ssi-schaefer-asia.com/storage-distribution/case-studies-from-all-asia/mobile-racking-systems/charoen-pokphand-foods-public-co-ltd.html, 10/05/2013.

ภาคผนวก ก.
ตารางพารามิเตอร์
สำหรับ SPA อัลกอริทึม

ตารางที่ ก.1 Earth periodic terms

Term	Row Number	A	B	C
L0	0	175347046	0	0
	1	3341656	4.6692568	6283.07585
	2	34894	4.6261	12566.1517
	3	3497	2.7441	5753.3849
	4	3418	2.8289	3.5231
	5	3136	3.6277	77713.7715
	6	2676	4.4181	7860.4194
	7	2343	6.1352	3930.2097
	8	1324	0.7425	11506.7698
	9	1273	2.0371	529.691
	10	1199	1.1096	1577.3435
	11	990	5.233	5884.927
	12	902	2.045	26.298
	13	857	3.508	398.149
	14	780	1.179	5223.694
	15	753	2.533	5507.553
	16	505	4.583	18849.228
	17	492	4.205	775.523
	18	357	2.92	0.067
	19	317	5.849	11790.629
	20	284	1.899	796.298
	21	271	0.315	10977.079
	22	243	0.345	5486.778
	23	206	4.806	2544.314
	24	205	1.869	5573.143
	25	202	2.458	6069.777
	26	156	0.833	213.299
	27	132	3.411	2942.463
	28	126	1.083	20.775
	29	115	0.645	0.98
	30	103	0.636	4694.003
	31	102	0.976	15720.839
	32	102	4.267	7.114
	33	99	6.21	2146.17
	34	98	0.68	155.42
	35	86	5.98	161000.69
	36	85	1.3	6275.96
	37	85	3.67	71430.7
	38	80	1.81	17260.15
	39	79	3.04	12036.46

ตารางที่ ก.2 Earth periodic terms (ต่อ)

Term	Row Number	A	B	C
	40	75	1.76	5088.63
	41	74	3.5	3154.69
	42	74	4.68	801.82
	43	70	0.83	9437.76
	44	62	3.98	8827.39
	45	61	1.82	7084.9
	46	57	2.78	6286.6
	47	56	4.39	14143.5
	48	56	3.47	6279.55
	49	52	0.19	12139.55
	50	52	1.33	1748.02
	51	51	0.28	5856.48
	52	49	0.49	1194.45
	53	41	5.37	8429.24
	54	41	2.4	19651.05
	55	39	6.17	10447.39
	56	37	6.04	10213.29
	57	37	2.57	1059.38
	58	36	1.71	2352.87
	59	36	1.78	6812.77
	60	33	0.59	17789.85
	61	30	0.44	83996.85
	62	30	2.74	1349.87
	63	25	3.16	4690.48
L1	0	628331966747	0	0
	1	206059	2.678235	6283.07585
	2	4303	2.6351	12566.1517
	3	425	1.59	3.523
	4	119	5.796	26.298
	5	109	2.966	1577.344
	6	93	2.59	18849.23
	7	72	1.14	529.69
	8	68	1.87	398.15
	9	67	4.41	5507.55
	10	59	2.89	5223.69
	11	56	2.17	155.42
	12	45	0.4	796.3
	13	36	0.47	775.52
	14	29	2.65	7.11
	15	21	5.34	0.98

ตารางที่ ก.3 Earth periodic terms (ต่อ)

Term	Row Number	A	B	C
	15	21	5.34	0.98
	16	19	1.85	5486.78
	17	19	4.97	213.3
	18	17	2.99	6275.96
	19	16	0.03	2544.31
	20	16	1.43	2146.17
	21	15	1.21	10977.08
	22	12	2.83	1748.02
	23	12	3.26	5088.63
	24	12	5.27	1194.45
	25	12	2.08	4694
	26	11	0.77	553.57
	27	10	1.3	6286.6
	28	10	4.24	1349.87
	29	9	2.7	242.73
	30	9	5.64	951.72
	31	8	5.3	2352.87
	32	6	2.65	9437.76
	33	6	4.67	4690.48
L2	0	52919	0	0
	1	8720	1.0721	6283.0758
	2	309	0.867	12566.152
	3	27	0.05	3.52
	4	16	5.19	26.3
	5	16	3.68	155.42
	6	10	0.76	18849.23
	7	9	2.06	77713.77
	8	7	0.83	775.52
	9	5	4.66	1577.34
	10	4	1.03	7.11
	11	4	3.44	5573.14
	12	3	5.14	796.3
	13	3	6.05	5507.55
	14	3	1.19	242.73
	15	3	6.12	529.69
	16	3	0.31	398.15
	17	3	2.28	553.57
	18	2	4.38	5223.69
	19	2	3.75	0.98
L3	0	289	5.844	6283.076

ตารางที่ ก.4 Earth periodic terms (ต่อ)

Term	Row Number	A	B	C
	1	35	0	0
	2	17	5.49	12566.15
	3	3	5.2	155.42
	4	1	4.72	3.52
	5	1	5.3	18849.23
	6	1	5.97	242.73
L4	0	114	3.142	0
	1	8	4.13	6283.08
	2	1	3.84	12566.15
L5	0	1	3.14	0
B0	0	280	3.199	84334.662
	1	102	5.422	5507.553
	2	80	3.88	5223.69
	3	44	3.7	2352.87
	4	32	4	1577.34
B1	0	9	3.9	5507.55
	1	6	1.73	5223.69
R0	0	100013989	0	0
	1	1670700	3.0984635	6283.07585
	2	13956	3.05525	12566.1517
	3	3084	5.1985	77713.7715
	4	1628	1.1739	5753.3849
	5	1576	2.8469	7860.4194
	6	925	5.453	11506.77
	7	542	4.564	3930.21
	8	472	3.661	5884.927
	9	346	0.964	5507.553
	10	329	5.9	5223.694
	11	307	0.299	5573.143
	12	243	4.273	11790.629
	13	212	5.847	1577.344
	14	186	5.022	10977.079
	15	175	3.012	18849.228
	16	110	5.055	5486.778
	17	98	0.89	6069.78
	18	86	5.69	15720.84
	19	86	1.27	161000.69
	20	65	0.27	17260.15
	21	63	0.92	529.69
	22	57	2.01	83996.85

ตารางที่ ก.5 Earth periodic terms (ต่อ)

Term	Row Number	A	B	C
	22	57	2.01	83996.85
	23	56	5.24	71430.7
	24	49	3.25	2544.31
	25	47	2.58	775.52
	26	45	5.54	9437.76
	27	43	6.01	6275.96
	28	39	5.36	4694
	29	38	2.39	8827.39
	30	37	0.83	19651.05
	31	37	4.9	12139.55
	32	36	1.67	12036.46
	33	35	1.84	2942.46
	34	33	0.24	7084.9
	35	32	0.18	5088.63
	36	32	1.78	398.15
	37	28	1.21	6286.6
	38	28	1.9	6279.55
	39	26	4.59	10447.39
R1	0	103019	1.10749	6283.07585
	1	1721	1.0644	12566.1517
	2	702	3.142	0
	3	32	1.02	18849.23
	4	31	2.84	5507.55
	5	25	1.32	5223.69
	6	18	1.42	1577.34
	7	10	5.91	10977.08
	8	9	1.42	6275.96
	9	9	0.27	5486.78
R2	0	4359	5.7846	6283.0758
	1	124	5.579	12566.152
	2	12	3.14	0
	3	9	3.63	77713.77
	4	6	1.87	5573.14
	5	3	5.47	18849.23
R3	0	145	4.273	6283.076
	1	7	3.92	12566.15
R4	0	4	2.56	6283.08

ตารางที่ ก.6 Periodic terms for the nutation in longitude and obliquity

Coefficients for sin terms					Coefficients for $\Delta\vartheta$		Coefficients for $\Delta\epsilon$	
Y0	**Y1**	**Y2**	**Y3**	**Y4**	**a**	**b**	**c**	**d**
0	0	0	0	1	-171996	-174.2	92025	8.9
-2	0	0	2	2	-13187	-1.6	5736	-3.1
0	0	0	2	2	-2274	-0.2	977	-0.5
0	0	0	0	2	2062	0.2	-895	0.5
0	1	0	0	0	1426	-3.4	54	-0.1
0	0	1	0	0	712	0.1	-7	
-2	1	0	2	2	-517	1.2	224	-0.6
0	0	0	2	1	-386	-0.4	200	
0	0	1	2	2	-301		129	-0.1
-2	-1	0	2	2	217	-0.5	-95	0.3
-2	0	1	0	0	-158			
-2	0	0	2	1	129	0.1	-70	
0	0	-1	2	2	123		-53	
2	0	0	0	0	63			
0	0	1	0	1	63	0.1	-33	
2	0	-1	2	2	-59		26	
0	0	-1	0	1	-58	-0.1	32	
0	0	1	2	1	-51		27	
-2	0	2	0	0	48			
0	0	-2	2	1	46		-24	
2	0	0	2	2	-38		16	
0	0	2	2	2	-31		13	
0	0	2	0	0	29			
-2	0	1	2	2	29		-12	
0	0	0	2	0	26			
-2	0	0	2	0	-22			
0	0	-1	2	1	21		-10	
0	2	0	0	0	17	-0.1		
2	0	-1	0	1	16		-8	
-2	2	0	2	2	-16	0.1	7	
0	1	0	0	1	-15		9	
-2	0	1	0	1	-13		7	
0	-1	0	0	1	-12		6	
0	0	2	-2	0	11			
2	0	-1	2	1	-10		5	
2	0	1	2	2	-8		3	
0	1	0	2	2	7		-3	
-2	1	1	0	0	-7			
0	-1	0	2	2	-7		3	

ตารางที่ ก.7 Periodic terms for the nutation in longitude and obliquity (ต่อ)

Coefficients for sin terms					Coefficients for $\Delta\vartheta$		Coefficients for $\Delta\epsilon$	
Y0	**Y1**	**Y2**	**Y3**	**Y4**	**a**	**b**	**c**	**d**
2	0	0	2	1	-7		3	
2	0	1	0	0	6			
-2	0	2	2	2	6		-3	
-2	0	1	2	1	6		-3	
2	0	-2	0	1	-6		3	
2	0	0	0	1	-6		3	
0	-1	1	0	0	5			
-2	-1	0	2	1	-5		3	
-2	0	0	0	1	-5		3	
0	0	2	2	1	-5		3	
-2	0	2	0	1	4			
-2	1	0	2	1	4			
0	0	1	-2	0	4			
-1	0	1	0	0	-4			
-2	1	0	0	0	-4			
1	0	0	0	0	-4			
0	0	1	2	0	3			
0	0	-2	2	2	-3			
-1	-1	1	0	0	-3			
0	1	1	0	0	-3			
0	-1	1	2	2	-3			
2	-1	-1	2	2	-3			
0	0	3	2	2	-3			
2	-1	0	2	2	-3			

ภาคผนวก ข.
โค๊ดโปรแกรม

SPA Header file (SPA.h) [2]

```
#ifndef __solar_position_algorithm_header
#define __solar_position_algorithm_header

//enumeration for function codes to select desired final outputs from SPA
enum {
	SPA_ZA,		//calculate zenith and azimuth
	SPA_ZA_INC, //calculate zenith, azimuth, and incidence
	SPA_ZA_RTS, //calculate zenith, azimuth, and sun rise/transit/set values
	SPA_ALL,	//calculate all SPA output values
};

typedef struct
{
	//----------------------INPUT VALUES-----------------------
	int year;		// 4-digit year, valid range: -2000 to 6000, error code: 1
	int month;		// 2-digit month, valid range: 1 to 12, error code: 2
	int day;		// 2-digit day, valid range: 1 to 31, error code: 3
	int hour;		// Observer local hour, valid range: 0 to 24, error code: 4
	int minute;		// Observer local minute, valid range: 0 to 59, error code: 5
	int second;		// Observer local second, valid range: 0 to 59, error code: 6
	double delta_t;	// Difference between earth rotation time and terrestrial time
			// It is derived from observation only and is reported in this
			// bulletin: http://maia.usno.navy.mil/ser7/ser7.dat,
			// where delta_t = 32.184 + (TAI-UTC) + DUT1
			// valid range: -8000 to 8000 seconds, error code: 7
	double timezone;	// Observer time zone (negative west of Greenwich)
			// valid range: -12 to 12 hours, error code: 8
	double longitude;	// Observer longitude (negative west of Greenwich)
			// valid range: -180 to 180 degrees, error code: 9
	double latitude;	// Observer latitude (negative south of equator)
			// valid range: -90 to 90 degrees, error code: 10
	double elevation;	// Observer elevation [meters]
			// valid range: -6500000 or higher meters, error code: 11
	double pressure;	// Annual average local pressure [millibars]
			// valid range: 0 to 5000 millibars, error code: 12
	double temperature;	// Annual average local temperature [degrees Celsius]
			// valid range: -273 to 6000 degrees Celsius, error code; 13
	double slope;		// Surface slope (measured from the horizontal plane)
			// valid range: -360 to 360 degrees, error code: 14
	double azm_rotation; // Surface azimuth rotation (measured from south to projection of
			// surface normal on horizontal plane, negative west)
			// valid range: -360 to 360 degrees, error code: 15
	double atmos_refract;// Atmospheric refraction at sunrise and sunset (0.5667 deg
			//is typical valid range: -5 to 5 degrees, error code: 16
	int function;	// Switch to choose functions for desired output (from enumeration)

	//----------------Intermediate OUTPUT VALUES--------------------
	double jd; //Julian day
	double jc; //Julian century
	double jde; //Julian ephemeris day
	double jce; //Julian ephemeris century
```

```
        double jme; //Julian ephemeris millennium
        double l; //earth heliocentric longitude [degrees]
        double b; //earth heliocentric latitude [degrees]
        double r; //earth radius vector [Astronomical Units, AU]
        double theta; //geocentric longitude [degrees]
        double beta; //geocentric latitude [degrees]
        double x0; //mean elongation (moon-sun) [degrees]
        double x1; //mean anomaly (sun) [degrees]
        double x2; //mean anomaly (moon) [degrees]
        double x3; //argument latitude (moon) [degrees]
        double x4; //ascending longitude (moon) [degrees]
        double del_psi; //nutation longitude [degrees]
        double del_epsilon; //nutation obliquity [degrees]
        double epsilon0; //ecliptic mean obliquity [arc seconds]
        double epsilon; //ecliptic true obliquity [degrees]
        double del_tau; //aberration correction [degrees]
        double lamda; //apparent sun longitude [degrees]
        double nu0; //Greenwich mean sidereal time [degrees]
        double nu; //Greenwich sidereal time [degrees]
        double alpha; //geocentric sun right ascension [degrees]
        double delta; //geocentric sun declination [degrees]
        double h; //observer hour angle [degrees]
        double xi; //sun equatorial horizontal parallax [degrees]
        double del_alpha; //sun right ascension parallax [degrees]
        double delta_prime; //topocentric sun declination [degrees]
        double alpha_prime; //topocentric sun right ascension [degrees]
        double h_prime; //topocentric local hour angle [degrees]
        double e0; //topocentric elevation angle (uncorrected) [degrees]
        double del_e; //atmospheric refraction correction [degrees]
        double e; //topocentric elevation angle (corrected) [degrees]
        double eot; //equation of time [minutes]
        double srha; //sunrise hour angle [degrees]
        double ssha; //sunset hour angle [degrees]
        double sta; //sun transit altitude [degrees]

        //----------------------Final OUTPUT VALUES-----------------------
        double zenith; //topocentric zenith angle [degrees]
        double azimuth180; //topocentric azimuth angle (westward from south)
                                    // [-180 to 180 degrees]
        double azimuth; //topocentric azimuth angle (eastward from north) [ 0 to 360 degrees]
        double incidence; //surface incidence angle [degrees]
        double suntransit; //local sun transit time (or solar noon) [fractional hour]
        double sunrise; //local sunrise time (+/- 30 seconds) [fractional hour]
        double sunset; //local sunset time (+/- 30 seconds) [fractional hour]
} spa_data;

//Calculate SPA output values (in structure) based on input values passed in structure
int spa_calculate(spa_data *spa);

#endif
```

SPA Source file (SPA.c) [2]

```
#include <math.h>
#include "spa.h"
#define PI 3.14159265358979323846264338327950288419 71
#define SUN_RADIUS 0.26667
#define L_COUNT 6
#define B_COUNT 2
#define R_COUNT 5
#define Y_COUNT 63
#define L_MAX_SUBCOUNT 64
#define B_MAX_SUBCOUNT 5
#define R_MAX_SUBCOUNT 40
enum {TERM_A, TERM_B, TERM_C, TERM_COUNT};
enum {TERM_X0, TERM_X1, TERM_X2, TERM_X3, TERM_X4, TERM_X_COUNT};
enum {TERM_PSI_A, TERM_PSI_B, TERM_EPS_C, TERM_EPS_D, TERM_PE_COUNT};
enum {JD_MINUS, JD_ZERO, JD_PLUS, JD_COUNT};
enum {SUN_TRANSIT, SUN_RISE, SUN_SET, SUN_COUNT};
#define TERM_Y_COUNT TERM_X_COUNT
const int l_subcount[L_COUNT] = {64,34,20,7,3,1};
const int b_subcount[B_COUNT] = {5,2};
const int r_subcount[R_COUNT] = {40,10,6,2,1};
//////////////////////////////////////////////////
/// Earth Periodic Terms
//////////////////////////////////////////////////
const double L_TERMS[L_COUNT][L_MAX_SUBCOUNT][TERM_COUNT]=
{
        {
                {175347046.0,0,0},
                {3341656.0,4.6692568,6283.07585},
                {34894.0,4.6261,12566.1517},
                {3497.0,2.7441,5753.3849},
                {3418.0,2.8289,3.5231},
                {3136.0,3.6277,77713.7715},
                {2676.0,4.4181,7860.4194},
                {2343.0,6.1352,3930.2097},
                {1324.0,0.7425,11506.7698},
                {1273.0,2.0371,529.691},
                {1199.0,1.1096,1577.3435},
                {990,5.233,5884.927},
                {902,2.045,26.298},
                {857,3.508,398.149},
                {780,1.179,5223.694},
                {753,2.533,5507.553},
                {505,4.583,18849.228},
                {492,4.205,775.523},
                {357,2.92,0.067},
                {317,5.849,11790.629},
                {284,1.899,796.298},
                {271,0.315,10977.079},
                {243,0.345,5486.778},
                {206,4.806,2544.314},
                {205,1.869,5573.143},
                {202,2.458,6069.777},
```

```
        {156,0.833,213.299},
        {132,3.411,2942.463},
        {126,1.083,20.775},
        {115,0.645,0.98},
        {103,0.636,4694.003},
        {102,0.976,15720.839},
        {102,4.267,7.114},
        {99,6.21,2146.17},
        {98,0.68,155.42},
        {86,5.98,161000.69},
        {85,1.3,6275.96},
        {85,3.67,71430.7},
        {80,1.81,17260.15},
        {79,3.04,12036.46},
        {75,1.76,5088.63},
        {74,3.5,3154.69},
        {74,4.68,801.82},
        {70,0.83,9437.76},
        {62,3.98,8827.39},
        {61,1.82,7084.9},
        {57,2.78,6286.6},
        {56,4.39,14143.5},
        {56,3.47,6279.55},
        {52,0.19,12139.55},
        {52,1.33,1748.02},
        {51,0.28,5856.48},
        {49,0.49,1194.45},
        {41,5.37,8429.24},
        {41,2.4,19651.05},
        {39,6.17,10447.39},
        {37,6.04,10213.29},
        {37,2.57,1059.38},
        {36,1.71,2352.87},
        {36,1.78,6812.77},
        {33,0.59,17789.85},
        {30,0.44,83996.85},
        {30,2.74,1349.87},
        {25,3.16,4690.48}
},
{
        {628331966747.0,0,0},
        {206059.0,2.678235,6283.07585},
        {4303.0,2.6351,12566.1517},
        {425.0,1.59,3.523},
        {119.0,5.796,26.298},
        {109.0,2.966,1577.344},
        {93,2.59,18849.23},
        {72,1.14,529.69},
        {68,1.87,398.15},
        {67,4.41,5507.55},
        {59,2.89,5223.69},
        {56,2.17,155.42},
        {45,0.4,796.3},
        {36,0.47,775.52},
        {29,2.65,7.11},
```

```
        {21,5.34,0.98},
        {19,1.85,5486.78},
        {19,4.97,213.3},
        {17,2.99,6275.96},
        {16,0.03,2544.31},
        {16,1.43,2146.17},
        {15,1.21,10977.08},
        {12,2.83,1748.02},
        {12,3.26,5088.63},
        {12,5.27,1194.45},
        {12,2.08,4694},
        {11,0.77,553.57},
        {10,1.3,6286.6},
        {10,4.24,1349.87},
        {9,2.7,242.73},
        {9,5.64,951.72},
        {8,5.3,2352.87},
        {6,2.65,9437.76},
        {6,4.67,4690.48}
},
{
        {52919.0,0,0},
        {8720.0,1.0721,6283.0758},
        {309.0,0.867,12566.152},
        {27,0.05,3.52},
        {16,5.19,26.3},
        {16,3.68,155.42},
        {10,0.76,18849.23},
        {9,2.06,77713.77},
        {7,0.83,775.52},
        {5,4.66,1577.34},
        {4,1.03,7.11},
        {4,3.44,5573.14},
        {3,5.14,796.3},
        {3,6.05,5507.55},
        {3,1.19,242.73},
        {3,6.12,529.69},
        {3,0.31,398.15},
        {3,2.28,553.57},
        {2,4.38,5223.69},
        {2,3.75,0.98}
},
{
        {289.0,5.844,6283.076},
        {35,0,0},
        {17,5.49,12566.15},
        {3,5.2,155.42},
        {1,4.72,3.52},
        {1,5.3,18849.23},
        {1,5.97,242.73}
},
{
        {114.0,3.142,0},
        {8,4.13,6283.08},
        {1,3.84,12566.15}
```

```
		},
		{
			{1,3.14,0}
		}
 };

const double B_TERMS[B_COUNT][B_MAX_SUBCOUNT][TERM_COUNT]=
{
		{
			{280.0,3.199,84334.662},
			{102.0,5.422,5507.553},
			{80,3.88,5223.69},
			{44,3.7,2352.87},
			{32,4,1577.34}
		},
		{
			{9,3.9,5507.55},
			{6,1.73,5223.69}
		}
};

const double R_TERMS[R_COUNT][R_MAX_SUBCOUNT][TERM_COUNT]=
{
		{
			{100013989.0,0,0},
			{1670700.0,3.0984635,6283.07585},
			{13956.0,3.05525,12566.1517},
			{3084.0,5.1985,77713.7715},
			{1628.0,1.1739,5753.3849},
			{1576.0,2.8469,7860.4194},
			{925.0,5.453,11506.77},
			{542.0,4.564,3930.21},
			{472.0,3.661,5884.927},
			{346.0,0.964,5507.553},
			{329.0,5.9,5223.694},
			{307.0,0.299,5573.143},
			{243.0,4.273,11790.629},
			{212.0,5.847,1577.344},
			{186.0,5.022,10977.079},
			{175.0,3.012,18849.228},
			{110.0,5.055,5486.778},
			{98,0.89,6069.78},
			{86,5.69,15720.84},
			{86,1.27,161000.69},
			{65,0.27,17260.15},
			{63,0.92,529.69},
			{57,2.01,83996.85},
			{56,5.24,71430.7},
			{49,3.25,2544.31},
			{47,2.58,775.52},
			{45,5.54,9437.76},
			{43,6.01,6275.96},
			{39,5.36,4694},
			{38,2.39,8827.39},
			{37,0.83,19651.05},
```

```
                {37,4.9,12139.55},
                {36,1.67,12036.46},
                {35,1.84,2942.46},
                {33,0.24,7084.9},
                {32,0.18,5088.63},
                {32,1.78,398.15},
                {28,1.21,6286.6},
                {28,1.9,6279.55},
                {26,4.59,10447.39}
        },
        {
                {103019.0,1.10749,6283.07585},
                {1721.0,1.0644,12566.1517},
                {702.0,3.142,0},
                {32,1.02,18849.23},
                {31,2.84,5507.55},
                {25,1.32,5223.69},
                {18,1.42,1577.34},
                {10,5.91,10977.08},
                {9,1.42,6275.96},
                {9,0.27,5486.78}
        },
        {
                {4359.0,5.7846,6283.0758},
                {124.0,5.579,12566.152},
                {12,3.14,0},
                {9,3.63,77713.77},
                {6,1.87,5573.14},
                {3,5.47,18849.23}
        },
        {
                {145.0,4.273,6283.076},
                {7,3.92,12566.15}
        },
        {
                {4,2.56,6283.08}
        }
};

//////////////////////////////////////////////////////////////
/// Periodic Terms for the nutation in longitude and obliquity
//////////////////////////////////////////////////////////////

const int Y_TERMS[Y_COUNT][TERM_Y_COUNT]=
{
        {0,0,0,0,1},
        {-2,0,0,2,2},
        {0,0,0,2,2},
        {0,0,0,0,2},
        {0,1,0,0,0},
        {0,0,1,0,0},
        {-2,1,0,2,2},
        {0,0,0,2,1},
        {0,0,1,2,2},
        {-2,-1,0,2,2},
```

```
        {-2,0,1,0,0},
        {-2,0,0,2,1},
        {0,0,-1,2,2},
        {2,0,0,0,0},
        {0,0,1,0,1},
        {2,0,-1,2,2},
        {0,0,-1,0,1},
        {0,0,1,2,1},
        {-2,0,2,0,0},
        {0,0,-2,2,1},
        {2,0,0,2,2},
        {0,0,2,2,2},
        {0,0,2,0,0},
        {-2,0,1,2,2},
        {0,0,0,2,0},
        {-2,0,0,2,0},
        {0,0,-1,2,1},
        {0,2,0,0,0},
        {2,0,-1,0,1},
        {-2,2,0,2,2},
        {0,1,0,0,1},
        {-2,0,1,0,1},
        {0,-1,0,0,1},
        {0,0,2,-2,0},
        {2,0,-1,2,1},
        {2,0,1,2,2},
        {0,1,0,2,2},
        {-2,1,1,0,0},
        {0,-1,0,2,2},
        {2,0,0,2,1},
        {2,0,1,0,0},
        {-2,0,2,2,2},
        {-2,0,1,2,1},
        {2,0,-2,0,1},
        {2,0,0,0,1},
        {0,-1,1,0,0},
        {-2,-1,0,2,1},
        {-2,0,0,0,1},
        {0,0,2,2,1},
        {-2,0,2,0,1},
        {-2,1,0,2,1},
        {0,0,1,-2,0},
        {-1,0,1,0,0},
        {-2,1,0,0,0},
        {1,0,0,0,0},
        {0,0,1,2,0},
        {0,0,-2,2,2},
        {-1,-1,1,0,0},
        {0,1,1,0,0},
        {0,-1,1,2,2},
        {2,-1,-1,2,2},
        {0,0,3,2,2},
        {2,-1,0,2,2},
};
```

```
const double PE_TERMS[Y_COUNT][TERM_PE_COUNT]=
{
	{-171996,-174.2,92025,8.9},
	{-13187,-1.6,5736,-3.1},
	{-2274,-0.2,977,-0.5},
	{2062,0.2,-895,0.5},
	{1426,-3.4,54,-0.1},
	{712,0.1,-7,0},
	{-517,1.2,224,-0.6},
	{-386,-0.4,200,0},
	{-301,0,129,-0.1},
	{217,-0.5,-95,0.3},
	{-158,0,0,0},
	{129,0.1,-70,0},
	{123,0,-53,0},
	{63,0,0,0},
	{63,0.1,-33,0},
	{-59,0,26,0},
	{-58,-0.1,32,0},
	{-51,0,27,0},
	{48,0,0,0},
	{46,0,-24,0},
	{-38,0,16,0},
	{-31,0,13,0},
	{29,0,0,0},
	{29,0,-12,0},
	{26,0,0,0},
	{-22,0,0,0},
	{21,0,-10,0},
	{17,-0.1,0,0},
	{16,0,-8,0},
	{-16,0.1,7,0},
	{-15,0,9,0},
	{-13,0,7,0},
	{-12,0,6,0},
	{11,0,0,0},
	{-10,0,5,0},
	{-8,0,3,0},
	{7,0,-3,0},
	{-7,0,0,0},
	{-7,0,3,0},
	{-7,0,3,0},
	{6,0,0,0},
	{6,0,-3,0},
	{6,0,-3,0},
	{-6,0,3,0},
	{-6,0,3,0},
	{5,0,0,0},
	{-5,0,3,0},
	{-5,0,3,0},
	{-5,0,3,0},
	{4,0,0,0},
	{4,0,0,0},
	{4,0,0,0},
	{-4,0,0,0},
```

```
        {-4,0,0,0},
        {-4,0,0,0},
        {3,0,0,0},
        {-3,0,0,0},
        {-3,0,0,0},
        {-3,0,0,0},
        {-3,0,0,0},
        {-3,0,0,0},
        {-3,0,0,0},
        {-3,0,0,0},
};

////////////////////////////////////////////
double rad2deg(double radians)
{
        return (180.0/PI)*radians;
}

double deg2rad(double degrees)
{
        return (PI/180.0)*degrees;
}

double limit_degrees(double degrees)
{
        double limited;
        degrees /= 360.0;
        limited = 360.0*(degrees-floor(degrees));
        if (limited < 0) limited += 360.0;
        return limited;
}

double limit_degrees180pm(double degrees)
{
        double limited;
        degrees /= 360.0;
        limited = 360.0*(degrees-floor(degrees));
        if (limited < -180.0) limited += 360.0;
        else if (limited > 180.0) limited -= 360.0;
        return limited;
}

double limit_degrees180(double degrees)
{
        double limited;
        degrees /= 180.0;
        limited = 180.0*(degrees-floor(degrees));
        if (limited < 0) limited += 180.0;
        return limited;
}

double limit_zero2one(double value)
{
        double limited;
        limited = value - floor(value);
```

```
	if (limited < 0) limited += 1.0;
	return limited;
}

double limit_minutes(double minutes)
{
	double limited=minutes;
	if (limited < -20.0) limited += 1440.0;
	else if (limited > 20.0) limited -= 1440.0;
	return limited;
}

double dayfrac_to_local_hr(double dayfrac, double timezone)
{
	return 24.0*limit_zero2one(dayfrac + timezone/24.0);
}

double third_order_polynomial(double a, double b, double c, double d, double x)
{
	return ((a*x + b)*x + c)*x + d;
}

///////////////////////////////////////////////////////////////////////////////////////
int validate_inputs(spa_data *spa)
{
	if ((spa->year < -2000) || (spa->year > 6000)) return 1;
	if ((spa->month < 1 ) || (spa->month > 12 )) return 2;
	if ((spa->day < 1 ) || (spa->day > 31 )) return 3;
	if ((spa->hour < 0 ) || (spa->hour > 24 )) return 4;
	if ((spa->minute < 0 ) || (spa->minute > 59 )) return 5;
	if ((spa->second < 0 ) || (spa->second > 59 )) return 6;
	if ((spa->pressure < 0 ) || (spa->pressure > 5000)) return 12;
	if ((spa->temperature <= -273) || (spa->temperature > 6000)) return 13;
	if ((spa->hour == 24 ) && (spa->minute > 0 )) return 5;
	if ((spa->hour == 24 ) && (spa->second > 0 )) return 6;
	if (fabs(spa->delta_t) > 8000 ) return 7;
	if (fabs(spa->timezone) > 12 ) return 8;
	if (fabs(spa->longitude) > 180 ) return 9;
	if (fabs(spa->latitude) > 90 ) return 10;
	if (fabs(spa->atmos_refract) > 5 ) return 16;
	if ( spa->elevation < -6500000) return 11;
	if ((spa->function == SPA_ZA_INC) || (spa->function == SPA_ALL))
	{
		if (fabs(spa->slope) > 360) return 14;
		if (fabs(spa->azm_rotation) > 360) return 15;
	}
	return 0;
}

///////////////////////////////////////////////////////////////////////////////////////
double julian_day (int year, int month, int day, int hour, int minute, int second, double tz)
{
	double day_decimal, julian_day, a;
	day_decimal = day + (hour - tz + (minute + second/60.0)/60.0)/24.0;
	if (month < 3) {
```

```
		month += 12;
		year--;
	}
	julian_day = floor(365.25*(year+4716.0)) + floor(30.6001*(month+1)) +
                                                        day_decimal - 1524.5;
	if (julian_day > 2299160.0) {
		a = floor(year/100);
		julian_day += (2 - a + floor(a/4));
	}
	return julian_day;
}

double julian_century(double jd)
{
	return (jd-2451545.0)/36525.0;
}

double julian_ephemeris_day(double jd, double delta_t)
{
	return jd+delta_t/86400.0;
}

double julian_ephemeris_century(double jde)
{
	return (jde - 2451545.0)/36525.0;
}

double julian_ephemeris_millennium(double jce)
{
	return (jce/10.0);
}

double earth_periodic_term_summation(const double terms[][TERM_COUNT],
                                                        int count, double jme)
{
	int i;
	double sum=0;
	for (i = 0; i < count; i++)
	sum += terms[i][TERM_A]*cos(terms[i][TERM_B]+terms[i][TERM_C]*jme);
	return sum;
}

double earth_values(double term_sum[], int count, double jme)
{
	int i;
	double sum=0;
	for (i = 0; i < count; i++)
	sum += term_sum[i]*pow(jme, i);
	sum /= 1.0e8;
	return sum;
}

double earth_heliocentric_longitude(double jme)
{
	double sum[L_COUNT];
```

```
        int i;
        for (i = 0; i < L_COUNT; i++)
        sum[i] = earth_periodic_term_summation(L_TERMS[i], l_subcount[i], jme);
        return limit_degrees(rad2deg(earth_values(sum, L_COUNT, jme)));
}

double earth_heliocentric_latitude(double jme)
{
        double sum[B_COUNT];
        int i;
        for (i = 0; i < B_COUNT; i++)
        sum[i] = earth_periodic_term_summation(B_TERMS[i], b_subcount[i], jme);
        return rad2deg(earth_values(sum, B_COUNT, jme));
}

double earth_radius_vector(double jme)
{
        double sum[R_COUNT];
        int i;
        for (i = 0; i < R_COUNT; i++)
        sum[i] = earth_periodic_term_summation(R_TERMS[i], r_subcount[i], jme);
        return earth_values(sum, R_COUNT, jme);
}

double geocentric_longitude(double l)
{
        double theta = l + 180.0;
        if (theta >= 360.0) theta -= 360.0;
        return theta;
}

double geocentric_latitude(double b)
{
        return -b;
}

double mean_elongation_moon_sun(double jce)
{
        return third_order_polynomial(1.0/189474.0, -0.0019142,
                                                    445267.11148, 297.85036, jce);
}
double mean_anomaly_sun(double jce)
{
        return third_order_polynomial(-1.0/300000.0, -0.0001603, 35999.05034, 357.52772, jce);
}

double mean_anomaly_moon(double jce)
{
        return third_order_polynomial(1.0/56250.0, 0.0086972, 477198.867398, 134.96298, jce);
}

double argument_latitude_moon(double jce)
{
        return third_order_polynomial(1.0/327270.0, -0.0036825,
                                                    483202.017538, 93.27191, jce);
```

```
}

double ascending_longitude_moon(double jce)
{
        return third_order_polynomial(1.0/450000.0, 0.0020708, -1934.136261, 125.04452, jce);
}

double xy_term_summation(int i, double x[TERM_X_COUNT])
{
        int j;
        double sum=0;
        for (j = 0; j < TERM_Y_COUNT; j++)
        sum += x[j]*Y_TERMS[i][j];
        return sum;
}

void nutation_longitude_and_obliquity(double jce, double x[TERM_X_COUNT],
                                                        double *del_psi, double *del_epsilon)
{
        int i;
        double xy_term_sum, sum_psi=0, sum_epsilon=0;
        for (i = 0; i < Y_COUNT; i++) {
        xy_term_sum = deg2rad(xy_term_summation(i, x));
        sum_psi += (PE_TERMS[i][TERM_PSI_A] +
                                jce*PE_TERMS[i][TERM_PSI_B])*sin(xy_term_sum);
        sum_epsilon += (PE_TERMS[i][TERM_EPS_C] +
                               jce*PE_TERMS[i][TERM_EPS_D])*cos(xy_term_sum);
        }
        *del_psi = sum_psi / 36000000.0;
        *del_epsilon = sum_epsilon / 36000000.0;
}

double ecliptic_mean_obliquity(double jme)
{
        double u = jme/10.0;
        return 84381.448 + u*(-4680.96 + u*(-1.55 + u*(1999.25 + u*(-51.38 + u*(-249.67 +
                                    u*( -39.05 + u*( 7.12 + u*( 27.87 + u*( 5.79 + u*2.45)))))))));
}

double ecliptic_true_obliquity(double delta_epsilon, double epsilon0)
{
        return delta_epsilon + epsilon0/3600.0;
}

double aberration_correction(double r)
{
        return -20.4898 / (3600.0*r);
}

double apparent_sun_longitude(double theta, double delta_psi, double delta_tau)
{
        return theta + delta_psi + delta_tau;
}
```

```
double greenwich_mean_sidereal_time (double jd, double jc)
{
	return limit_degrees(280.46061837 + 360.98564736629 * (jd - 2451545.0) +
                                                        jc*jc*(0.000387933 - jc/38710000.0));
}

double greenwich_sidereal_time (double nu0, double delta_psi, double epsilon)
{
return nu0 + delta_psi*cos(deg2rad(epsilon));
}

double geocentric_sun_right_ascension(double lamda, double epsilon, double beta)
{
	double lamda_rad = deg2rad(lamda);
	double epsilon_rad = deg2rad(epsilon);
	return limit_degrees(rad2deg(atan2(sin(lamda_rad)*cos(epsilon_rad) -
	tan(deg2rad(beta))*sin(epsilon_rad), cos(lamda_rad))));
}

double geocentric_sun_declination(double beta, double epsilon, double lamda)
{
	double beta_rad = deg2rad(beta);
	double epsilon_rad = deg2rad(epsilon);
	return rad2deg(asin(sin(beta_rad)*cos(epsilon_rad) +
	cos(beta_rad)*sin(epsilon_rad)*sin(deg2rad(lamda))));
}

double observer_hour_angle(double nu, double longitude, double alpha_deg)
{
	return limit_degrees(nu + longitude - alpha_deg);
}

double sun_equatorial_horizontal_parallax(double r)
{
	return 8.794 / (3600.0 * r);
}

void sun_right_ascension_parallax_and_topocentric_dec(double latitude, double elevation,
		double xi, double h, double delta, double *delta_alpha, double *delta_prime)
{
	double delta_alpha_rad;
	double lat_rad = deg2rad(latitude);
	double xi_rad = deg2rad(xi);
	double h_rad = deg2rad(h);
	double delta_rad = deg2rad(delta);
	double u = atan(0.99664719 * tan(lat_rad));
	double y = 0.99664719 * sin(u) + elevation*sin(lat_rad)/6378140.0;
	double x = cos(u) + elevation*cos(lat_rad)/6378140.0;
	delta_alpha_rad = atan2( - x*sin(xi_rad) *sin(h_rad),
                                            cos(delta_rad) - x*sin(xi_rad) *cos(h_rad));
	*delta_prime = rad2deg(atan2((sin(delta_rad) - y*sin(xi_rad))*cos(delta_alpha_rad),
                                                cos(delta_rad) - x*sin(xi_rad) *cos(h_rad)));
	*delta_alpha = rad2deg(delta_alpha_rad);
}
```

```
double topocentric_sun_right_ascension(double alpha_deg, double delta_alpha)
{
	return alpha_deg + delta_alpha;
}

double topocentric_local_hour_angle(double h, double delta_alpha)
{
	return h - delta_alpha;
}

double topocentric_elevation_angle(double latitude, double delta_prime, double h_prime)
{
	double lat_rad = deg2rad(latitude);
	double delta_prime_rad = deg2rad(delta_prime);
	return rad2deg(asin(sin(lat_rad)*sin(delta_prime_rad) +
				cos(lat_rad)*cos(delta_prime_rad) * cos(deg2rad(h_prime))));
}

double atmospheric_refraction_correction(double pressure, double temperature,
                                                        double atmos_refract, double e0)
{
	double del_e = 0;
	if (e0 >= -1*(SUN_RADIUS + atmos_refract))
	del_e = (pressure / 1010.0) * (283.0 / (273.0 + temperature)) *
	1.02 / (60.0 * tan(deg2rad(e0 + 10.3/(e0 + 5.11))));
	return del_e;
}

double topocentric_elevation_angle_corrected(double e0, double delta_e)
{
	return e0 + delta_e;
}

double topocentric_zenith_angle(double e)
{
	return 90.0 - e;
}

double topocentric_azimuth_angle_neg180_180(double h_prime, double latitude,
                                                                double delta_prime)
{
	double h_prime_rad = deg2rad(h_prime);
	double lat_rad = deg2rad(latitude);
	return rad2deg(atan2(sin(h_prime_rad),
	cos(h_prime_rad)*sin(lat_rad) -
	tan(deg2rad(delta_prime))*cos(lat_rad)));
}

double topocentric_azimuth_angle_zero_360(double azimuth180)
{
	return azimuth180 + 180.0;
}
```

```
double surface_incidence_angle(double zenith, double azimuth180,
                                        double azm_rotation, double slope)
{
        double zenith_rad = deg2rad(zenith);
        double slope_rad = deg2rad(slope);
        return rad2deg(acos(cos(zenith_rad)*cos(slope_rad) +
        sin(slope_rad )*sin(zenith_rad) * cos(deg2rad(azimuth180 -
        azm_rotation))));
}
double sun_mean_longitude(double jme)
{
        return limit_degrees(280.4664567 + jme*(360007.6982779 + jme*(0.03032028 +
                        jme*(1/49931.0 + jme*(-1/15300.0 + jme*(-1/2000000.0))))));
}

double eot(double m, double alpha, double del_psi, double epsilon)
{
        return limit_minutes(4.0*(m - 0.0057183 - alpha + del_psi*cos(deg2rad(epsilon))));
}

double approx_sun_transit_time(double alpha_zero, double longitude, double nu)
{
        return (alpha_zero - longitude - nu) / 360.0;
}

double sun_hour_angle_at_rise_set(double latitude, double delta_zero, double h0_prime)
{
        double h0 = -99999;
        double latitude_rad = deg2rad(latitude);
        double delta_zero_rad = deg2rad(delta_zero);
        double argument = (sin(deg2rad(h0_prime)) - sin(latitude_rad)*sin(delta_zero_rad)) /
                                                (cos(latitude_rad)*cos(delta_zero_rad));
        if (fabs(argument) <= 1) h0 = limit_degrees180(rad2deg(acos(argument)));
        return h0;
}

void approx_sun_rise_and_set(double *m_rts, double h0)
{
        double h0_dfrac = h0/360.0;
        m_rts[SUN_RISE] = limit_zero2one(m_rts[SUN_TRANSIT] - h0_dfrac);
        m_rts[SUN_SET] = limit_zero2one(m_rts[SUN_TRANSIT] + h0_dfrac);
        m_rts[SUN_TRANSIT] = limit_zero2one(m_rts[SUN_TRANSIT]);
}

double rts_alpha_delta_prime(double *ad, double n)
{
        double a = ad[JD_ZERO] - ad[JD_MINUS];
        double b = ad[JD_PLUS] - ad[JD_ZERO];
        if (fabs(a) >= 2.0) a = limit_zero2one(a);
        if (fabs(b) >= 2.0) b = limit_zero2one(b);
        return ad[JD_ZERO] + n * (a + b + (b-a)*n)/2.0;
}
```

```
double rts_sun_altitude(double latitude, double delta_prime, double h_prime)
{
	double latitude_rad = deg2rad(latitude);
	double delta_prime_rad = deg2rad(delta_prime);
	return rad2deg(asin(sin(latitude_rad)*sin(delta_prime_rad) +
	cos(latitude_rad)*cos(delta_prime_rad)*cos(deg2rad(h_prime))));
}

double sun_rise_and_set(double *m_rts, double *h_rts, double *delta_prime,
                                    double latitude,double *h_prime, double h0_prime, int sun)
{
	return m_rts[sun] + (h_rts[sun] - h0_prime) /
                        (360.0*cos(deg2rad(delta_prime[sun]))*cos(deg2rad(latitude))
                                                                        *sin(deg2rad(h_prime[sun])));
}

//////////////////////////////////////////////////////////////////////////////////////////
// Calculate required SPA parameters to get the right ascension (alpha) and declination (delta)
// Note: JD must be already calculated and in structure
//////////////////////////////////////////////////////////////////////////////////////////
void calculate_geocentric_sun_right_ascension_and_declination(spa_data *spa)
{
	double x[TERM_X_COUNT];
	spa->jc = julian_century(spa->jd);
	spa->jde = julian_ephemeris_day(spa->jd, spa->delta_t);
	spa->jce = julian_ephemeris_century(spa->jde);
	spa->jme = julian_ephemeris_millennium(spa->jce);
	spa->l = earth_heliocentric_longitude(spa->jme);
	spa->b = earth_heliocentric_latitude(spa->jme);
	spa->r = earth_radius_vector(spa->jme);
	spa->theta = geocentric_longitude(spa->l);
	spa->beta = geocentric_latitude(spa->b);
	x[TERM_X0] = spa->x0 = mean_elongation_moon_sun(spa->jce);
	x[TERM_X1] = spa->x1 = mean_anomaly_sun(spa->jce);
	x[TERM_X2] = spa->x2 = mean_anomaly_moon(spa->jce);
	x[TERM_X3] = spa->x3 = argument_latitude_moon(spa->jce);
	x[TERM_X4] = spa->x4 = ascending_longitude_moon(spa->jce);
	nutation_longitude_and_obliquity(spa->jce, x, &(spa->del_psi), &(spa->del_epsilon));
	spa->epsilon0 = ecliptic_mean_obliquity(spa->jme);
	spa->epsilon = ecliptic_true_obliquity(spa->del_epsilon, spa->epsilon0);
	spa->del_tau = aberration_correction(spa->r);
	spa->lamda = apparent_sun_longitude(spa->theta, spa->del_psi, spa->del_tau);
	spa->nu0 = greenwich_mean_sidereal_time (spa->jd, spa->jc);
	spa->nu = greenwich_sidereal_time (spa->nu0, spa->del_psi, spa->epsilon);
	spa->alpha = geocentric_sun_right_ascension(spa->lamda, spa->epsilon, spa->beta);
	spa->delta = geocentric_sun_declination(spa->beta, spa->epsilon, spa->lamda);
}

//////////////////////////////////////////////////////////////////////
// Calculate Equation of Time (EOT) and Sun Rise, Transit, & Set (RTS)
//////////////////////////////////////////////////////////////////////
void calculate_eot_and_sun_rise_transit_set(spa_data *spa)
{
	spa_data sun_rts = *spa;
	double nu, m, h0, n;
```

```
        double alpha[JD_COUNT], delta[JD_COUNT];
        double m_rts[SUN_COUNT], nu_rts[SUN_COUNT], h_rts[SUN_COUNT];
        double alpha_prime[SUN_COUNT], delta_prime[SUN_COUNT],
                                                        h_prime[SUN_COUNT];
        double h0_prime = -1*(SUN_RADIUS + spa->atmos_refract);
        int i;
        m = sun_mean_longitude(spa->jme);
        spa->eot = eot(m, spa->alpha, spa->del_psi, spa->epsilon);
        sun_rts.hour = sun_rts.minute = sun_rts.second = sun_rts.timezone = 0;
        sun_rts.jd = julian_day (sun_rts.year, sun_rts.month, sun_rts.day,
        sun_rts.hour, sun_rts.minute, sun_rts.second, sun_rts.timezone);
        calculate_geocentric_sun_right_ascension_and_declination(&sun_rts);
        nu = sun_rts.nu;
        sun_rts.delta_t = 0;
        sun_rts.jd--;

        for (i = 0; i < JD_COUNT; i++) {
                calculate_geocentric_sun_right_ascension_and_declination(&sun_rts);
                alpha[i] = sun_rts.alpha;
                delta[i] = sun_rts.delta;
                sun_rts.jd++;
        }

        m_rts[SUN_TRANSIT] = approx_sun_transit_time(alpha[JD_ZERO],
                                                        spa->longitude, nu);
        h0 = sun_hour_angle_at_rise_set(spa->latitude, delta[JD_ZERO], h0_prime);

        if (h0 >= 0) {
                approx_sun_rise_and_set(m_rts, h0);
                for (i = 0; i < SUN_COUNT; i++) {
                        nu_rts[i] = nu + 360.985647*m_rts[i];
                        n = m_rts[i] + spa->delta_t/86400.0;
                        alpha_prime[i] = rts_alpha_delta_prime(alpha, n);
                        delta_prime[i] = rts_alpha_delta_prime(delta, n);
                        h_prime[i] = limit_degrees180pm(nu_rts[i] + spa->longitude –
                                                                alpha_prime[i]);
                        h_rts[i] = rts_sun_altitude(spa->latitude, delta_prime[i], h_prime[i]);
                }

                spa->srha = h_prime[SUN_RISE];
                spa->ssha = h_prime[SUN_SET];
                spa->sta = h_rts[SUN_TRANSIT];
                spa->suntransit = dayfrac_to_local_hr(m_rts[SUN_TRANSIT] –
                                                        h_prime[SUN_TRANSIT] / 360.0,
                spa->timezone);
                spa->sunrise = dayfrac_to_local_hr(sun_rise_and_set(m_rts, h_rts, delta_prime,
                spa->latitude, h_prime, h0_prime, SUN_RISE), spa->timezone);
                spa->sunset = dayfrac_to_local_hr(sun_rise_and_set(m_rts, h_rts, delta_prime,
                spa->latitude, h_prime, h0_prime, SUN_SET), spa->timezone);
        }
        else spa->srha= spa->ssha= spa->sta= spa->suntransit=
                                        spa->sunrise= spa->sunset= -99999;
}
```

```
//////////////////////////////////////////////////////////////////////////////////
// Calculate all SPA parameters and put into structure
// Note: All inputs values (listed in header file) must already be in structure
//////////////////////////////////////////////////////////////////////////////////
int spa_calculate(spa_data *spa)
{
	int result;
	result = validate_inputs(spa);
	if (result == 0)
	{
		spa->jd = julian_day (spa->year, spa->month, spa->day,
		spa->hour, spa->minute, spa->second, spa->timezone);
		calculate_geocentric_sun_right_ascension_and_declination(spa);
		spa->h = observer_hour_angle(spa->nu, spa->longitude, spa->alpha);
		spa->xi = sun_equatorial_horizontal_parallax(spa->r);
		sun_right_ascension_parallax_and_topocentric_dec(spa->latitude,
		                                                 spa->elevation, spa->xi,
		spa->h, spa->delta, &(spa->del_alpha), &(spa->delta_prime));
		spa->alpha_prime = topocentric_sun_right_ascension(spa->alpha,
		                                                 spa->del_alpha);
		spa->h_prime = topocentric_local_hour_angle(spa->h, spa->del_alpha);
		spa->e0 = topocentric_elevation_angle(spa->latitude,
		                                      spa->delta_prime,spa->h_prime);
		spa->del_e = atmospheric_refraction_correction(spa->pressure,
		                           spa->temperature, spa->atmos_refract, spa->e0);
		spa->e = topocentric_elevation_angle_corrected(spa->e0, spa->del_e);
		spa->zenith = topocentric_zenith_angle(spa->e);
		spa->azimuth180 = topocentric_azimuth_angle_neg180_180(spa->h_prime,
		                                      spa->latitude, spa->delta_prime);
		spa->azimuth = topocentric_azimuth_angle_zero_360(spa->azimuth180);

		if ((spa->function == SPA_ZA_INC) || (spa->function == SPA_ALL))
			spa->incidence = surface_incidence_angle(spa->zenith,
			  spa->azimuth180, spa->azm_rotation, spa->slope);
		if ((spa->function == SPA_ZA_RTS) || (spa->function == SPA_ALL))
		calculate_eot_and_sun_rise_transit_set(spa);
	}
	return result;
}
//////////////////////////////////////////////////////////////////////////////////
```

Arduino Source file

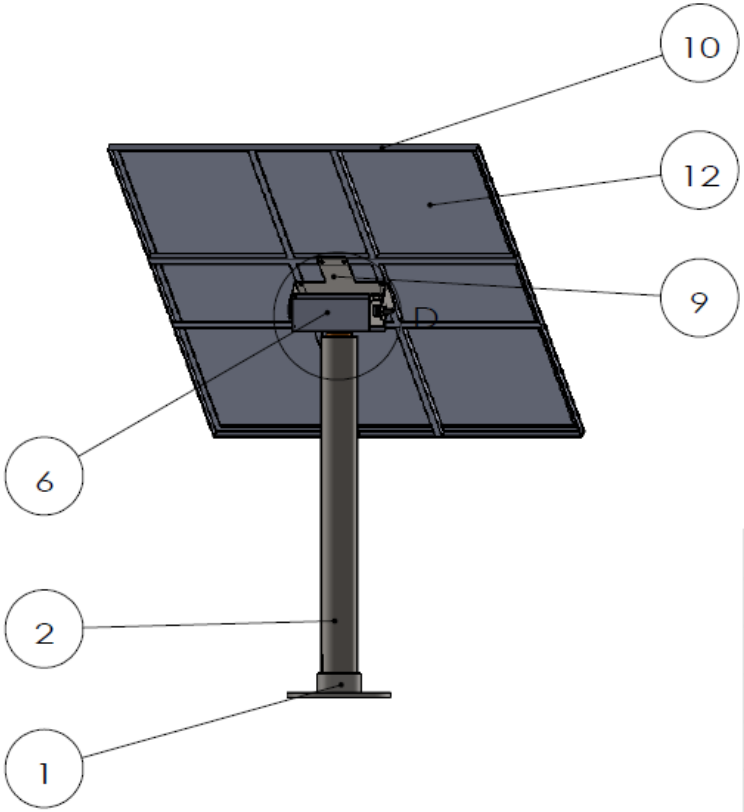
```
#include <math.h>
#include "SPA.h"
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 rtc;
int result;
int ALT_OUT = 13, AZI_OUT = 9, DirectionALT = 12, DirectionAZI = 8;
float mins, sec;
double    TarX =-19.5 ,TarY =0, TarZ =3.5, SVecX, SVecY, SVecZ, NormSVec,NormTVec,
          NSX, NSY, NSZ, NTX, NTY, NTZ, NormST, NX, NY, NZ,HeliALT, HeliAZI,
          DeltaALT, DeltaAZI, HeliALTOLD, HeliAZIOLD, A, B, PulseALT, PulseAZI;

void setup()
{
        Serial.begin(9600);
        Wire.begin();
        Wire1.begin();
        rtc.begin();
        pinMode(ALT_OUT, OUTPUT);
        pinMode(DirectionALT, OUTPUT);
        pinMode(AZI_OUT, OUTPUT);
        pinMode(DirectionAZI, OUTPUT);
}

void loop()
{
        DateTime now = rtc.now();

        Serial.println();
        Serial.print(now.year(), DEC);
        Serial.print('/');
        Serial.print(now.month(), DEC);
        Serial.print('/');
        Serial.print(now.day(), DEC);
        Serial.print(' ');
        Serial.print(now.hour(), DEC);
        Serial.print(':');
        Serial.print(now.minute(), DEC);
        Serial.print(':');
        Serial.print(now.second(), DEC);
        Serial.println();

        spa_data spa;

        spa.years  = now.year();
        spa.months  = now.month();
        spa.days  = now.day();
        spa.hours  = now.hour();
        spa.minutes  = now.minute();
        spa.seconds  = now.second();
        spa.timezone = 7.0;
```

```
spa.delta_ut1 = 0;
spa.delta_t  = 67;
spa.longitude = 100.724439;
spa.latitude = 14.036711;
spa.elevation = 7;
spa.pressure = 761.725;
spa.temperature = 29;
spa.slope  = 30;
spa.azm_rotation  = -10;
spa.atmos_refract = 0.5667;
spa.function = SPA_ALL;
result = spa_calculate(&spa);

if (result == 0)
{
        Serial.println("Sun Altitude&Azimuth angle are running");
        Serial.println();
}
else printf("SPA Error Code: %d\n", result);

 SVecX = cos((90-spa.zenith)*(6.2832/360))*cos(spa.azimuth*(6.2832/360));
 SVecY = cos((90-spa.zenith)*(6.2832/360))*sin(spa.azimuth*(6.2832/360));
 SVecZ = sin((90-spa.zenith)*(6.2832/360));
 NormSVec = sqrt((SVecX*SVecX)+(SVecY*SVecY)+(SVecZ*SVecZ));
 NSX = SVecX/NormSVec;
 NSY = SVecY/NormSVec;
 NSZ = SVecZ/NormSVec;
 NormTVec = sqrt((TarX*TarX)+(TarY*TarY)+(TarZ*TarZ));
 NTX = TarX/NormTVec;
 NTY = TarY/NormTVec;
 NTZ = TarZ/NormTVec;
 NormST = sqrt(((NTX+NSX)*(NTX+NSX))+((NTY+NSY)*(NTY+NSY))+
                                                ((NTZ+NSZ)*(NTZ+NSZ)));
  NX = (NTX+NSX)/NormST;
  NY = (NTY+NSY)/NormST;
  NZ = (NTZ+NSZ)/NormST;

  HeliALT = 90-(asin(NZ)*(360/6.2832));
  HeliAZI = (asin((NY)/(sqrt((NX*NX)+(NY*NY)))))*(360/6.2832);

  Serial.print("HeliALT (degree) = ");
  Serial.println(HeliALT);                //Show Heliostat's Altitude angle (degree)
  Serial.print("HeliAZI (degree) = ");
  Serial.println(HeliAZI);                //Show Heliostat's Azimuth angle (degree)
  Serial.println();

  DeltaALT = HeliALTOLD+A-HeliALT;
  DeltaAZI = HeliAZIOLD+B-HeliAZI;

  Serial.print("Delta_ALT (degree) = ");
  Serial.println(DeltaALT);         //Show differential of Heliostat's Altitude angle (degree)
  Serial.print("Delta_AZI (degree) = ");
  Serial.println(DeltaAZI);         //Show differential of Heliostat's Azimuth angle (degree)
  Serial.println();
```

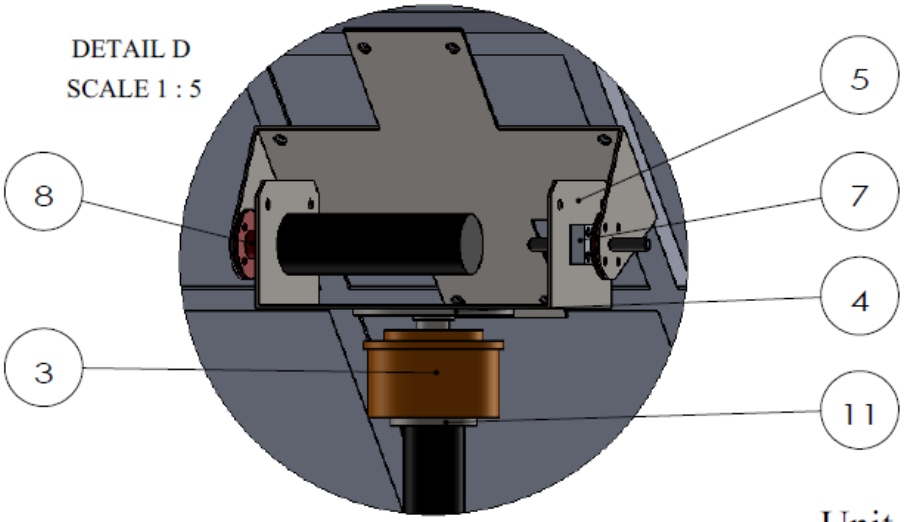
```
  HeliALTOLD = HeliALT;
  HeliAZIOLD = HeliAZI;

  Serial.print("HeliALT_OLD (degree) = ");
  Serial.println(HeliALTOLD);    //Show Actual Heliostat's Altitude angle (degree)
  Serial.print("HeliAZI_OLD (degree) = ");
  Serial.println(HeliAZIOLD);     //Show Actual Heliostat's Azimuth angle (degree)
  Serial.println();

  PulseALT = abs(DeltaALT)*(1112000/360);  //Count of Altitude Angle
  PulseAZI = abs(DeltaAZI)*(1112000/360);  //Count of Azimuth Angle

  Serial.print("Pulse of Altitude Angle = ");
  Serial.println(PulseALT);   //Show Count of Altitude Angle
  Serial.print("Pulse of Azimuth Angle = ");
  Serial.println(PulseAZI);    //Show Count of Azimuth Angle
  Serial.println();

 if (DeltaALT < 0)
 {
        for (int i = 0; i < PulseALT; i++)
        {
              digitalWrite(DirectionALT, LOW);
              digitalWrite(ALT_OUT, HIGH);
              delay(1);
              digitalWrite(ALT_OUT, LOW);
              delay(1);
        }
        Serial.println("Delta_ALT < 0 is running");
        Serial.println();
}
else
{
        for (int i = 0; i < PulseALT; i++)
        {
              digitalWrite(DirectionALT, HIGH);
              digitalWrite(ALT_OUT, HIGH);
              delay(1);
              digitalWrite(ALT_OUT, LOW);
              delay(1);
        }
        Serial.println("Delta_ALT > 0 is running");
        Serial.println();
 }
 if (DeltaAZI < 0)
 {
        for (int i = 0; i < PulseAZI; i++)
        {
              digitalWrite(DirectionAZI, HIGH);
              digitalWrite(AZI_OUT, HIGH);
              delay(1);
              digitalWrite(AZI_OUT, LOW);
              delay(1);
        }
```

```
                Serial.println("Delta_AZI < 0 is running");
                Serial.println();
        }
        else
        {
                for (int i = 0; i < PulseAZI; i++)
                {
                        digitalWrite(DirectionAZI, LOW);
                        digitalWrite(AZI_OUT, HIGH);
                        delay(1);
                        digitalWrite(AZI_OUT, LOW);
                        delay(1);
                }
                Serial.println("Delta_AZI > 0 is running");
                Serial.println();
        }
        delay(120000);
}
```

ภาคผนวก ค.
แบบโครงสร้าง



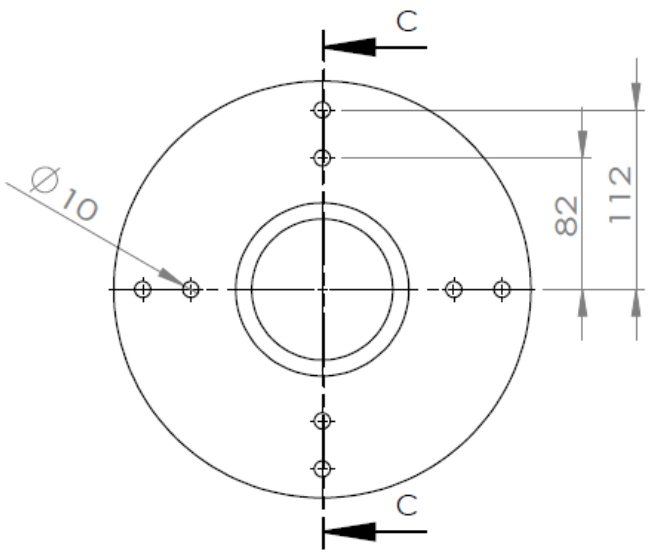
DETAIL D
SCALE 1 : 5

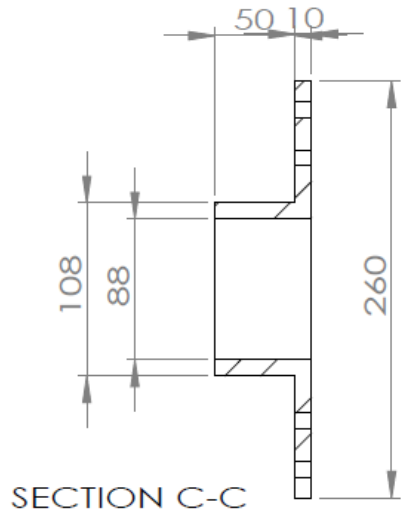


Unit : mm.

NO.	DRAWING NAME	NO.	DRAWING NAME
1	BASE	7	BUSH BEARING
2	COLUMN	8	BRACKET
3	BUSH	9	PLATE
4	SHAFT PLATE	10	MIRROR FLAME
5	MOTOR BASE	11	MOTOR SPACER
6	MOTOR COVER	12	MIRROR

NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI
DEPARTMENT	MECHANICAL ENGINEERING	
GROUP	54343 MEE	
CODE	115430431026-5	
SCALE 1:10	**DWG NAME :DEVELOPMENT OF SUN REFLECTING CONTROL SYSTEM BY MICROCONTROLLER**	PART NUMBER : -



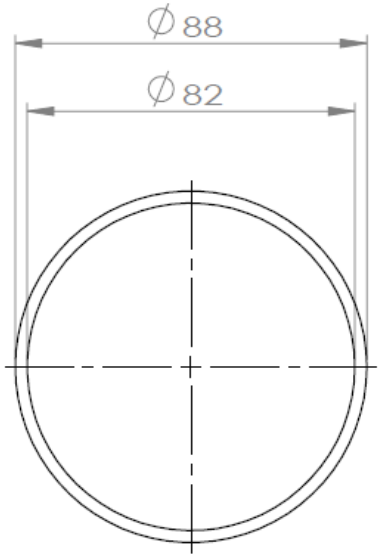


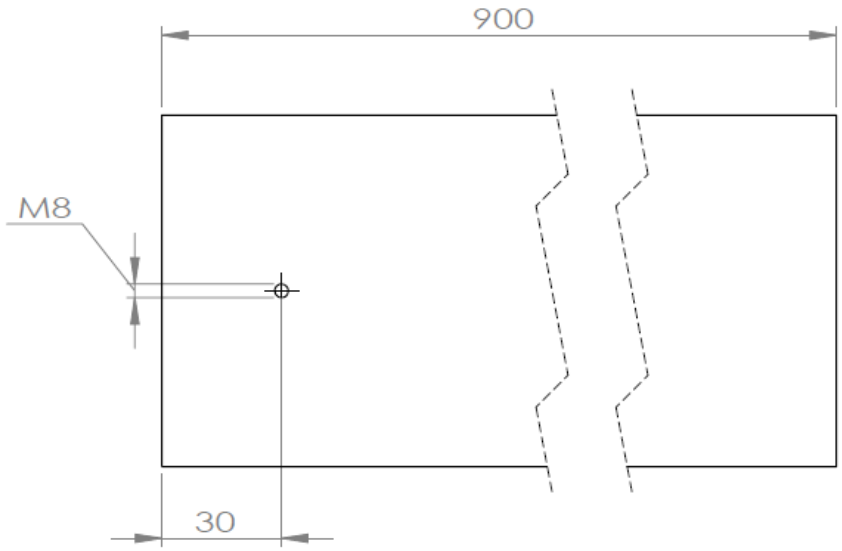
SECTION C-C

SCALE 1 : 5

Unit : mm.

MATERIAL: STEEL ST 37			NUMBER : 1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 1:2	DWG NAME : BASE	PART NUMBER : 1	



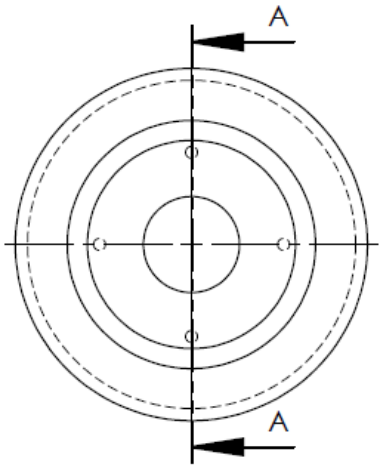


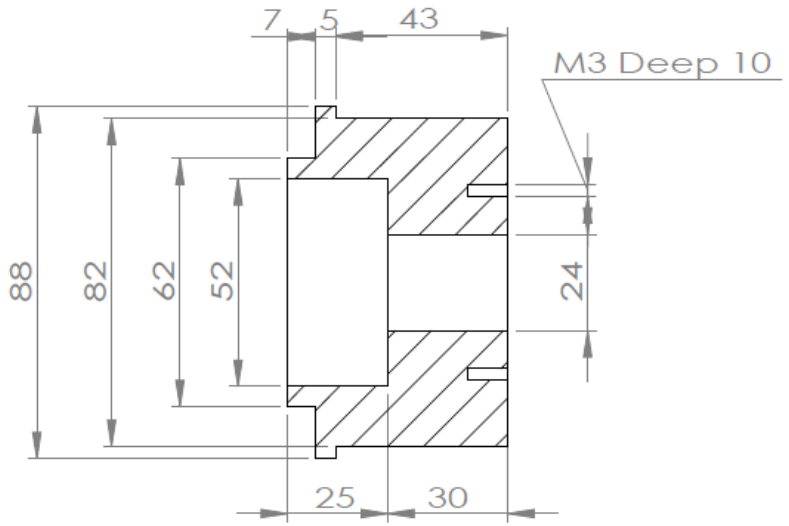
Unit : mm.

MATERIAL: STEEL ST 37			NUMBER :1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE :1:5	DWG NAME : COLUMN	PART NUMBER : 2	

SECTION A-A

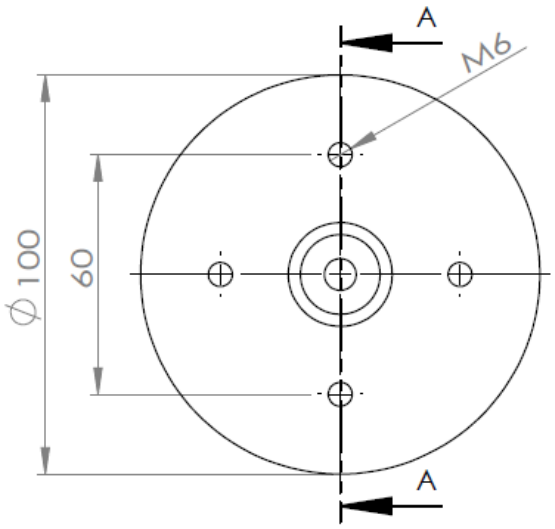
SCALE 1 : 2

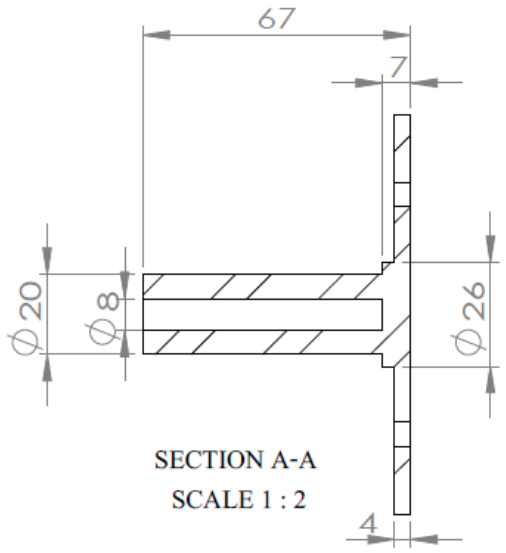




Unit : mm.

MATERIAL: ALUMINUM 6060			NUMBER : 1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 1:2	DWG NAME : BUSH	PART NUMBER : 3	

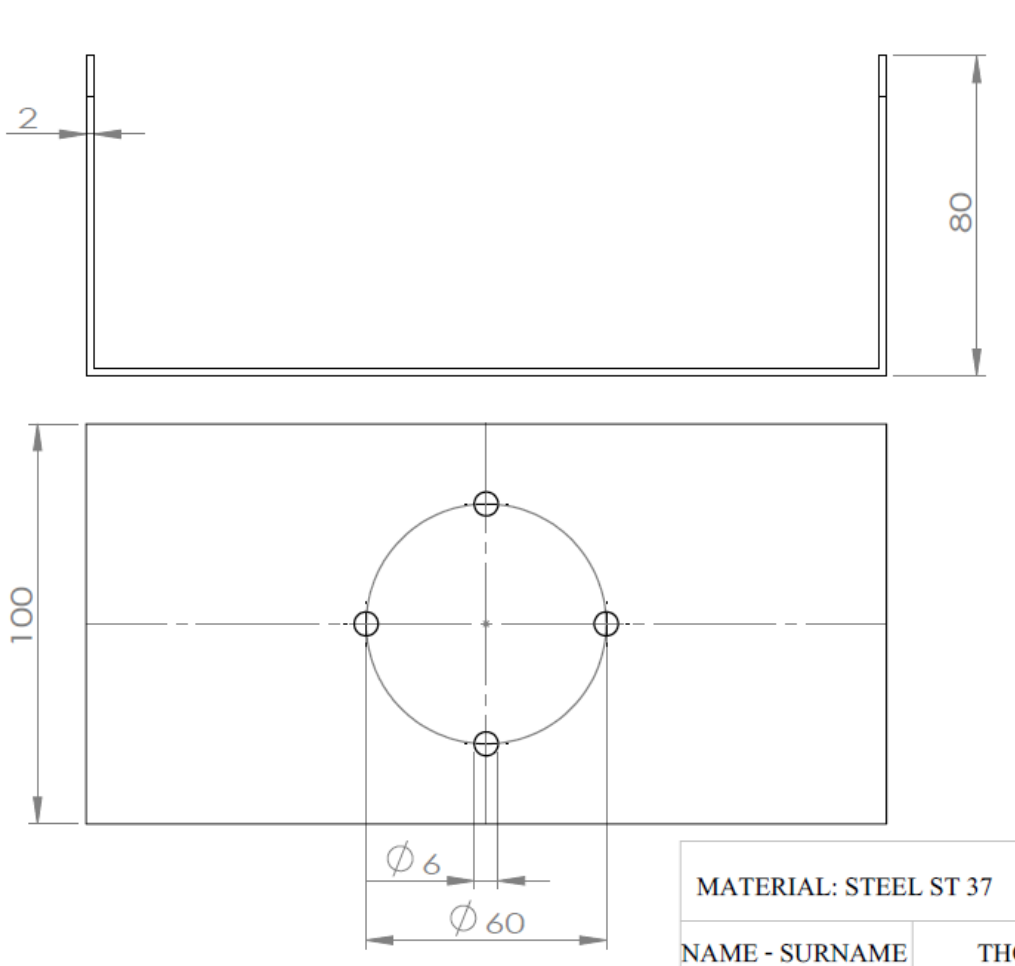


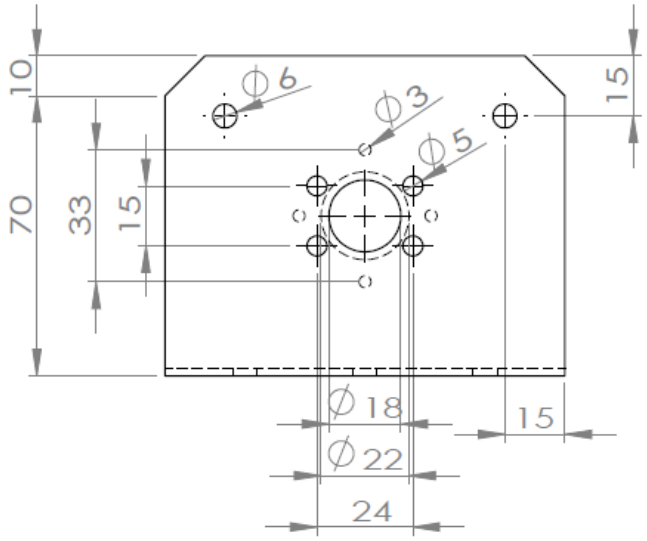


SECTION A-A
SCALE 1 : 2

Unit : mm.

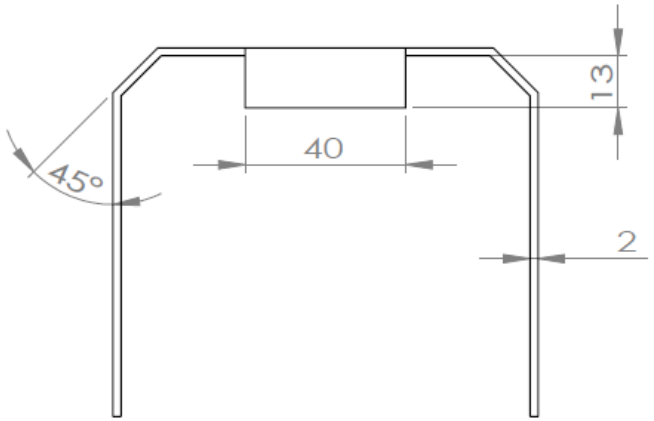
MATERIAL: ALUMNUM 6060			NUMBER :1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE :1:2	DWG NAME : SHAFT PLATE	PART NUMBER : 4	

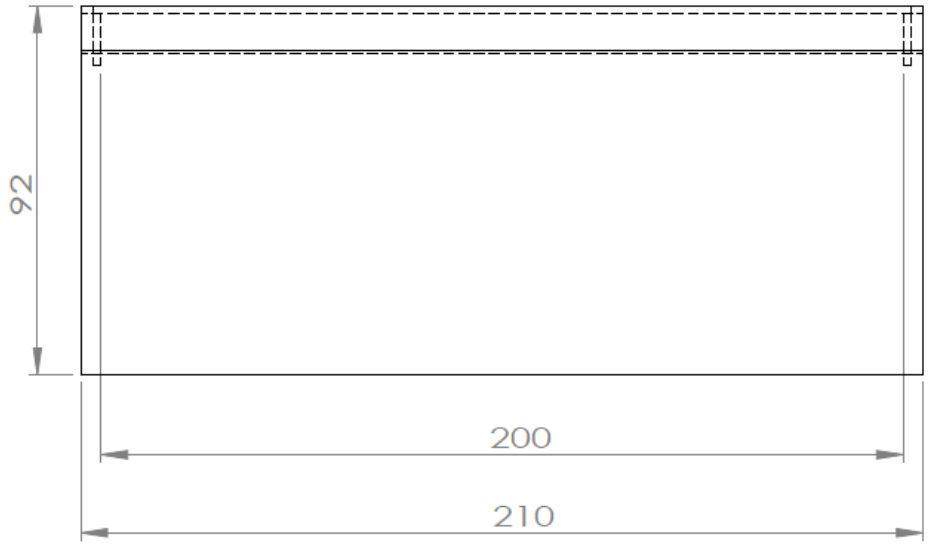




Unit : mm.

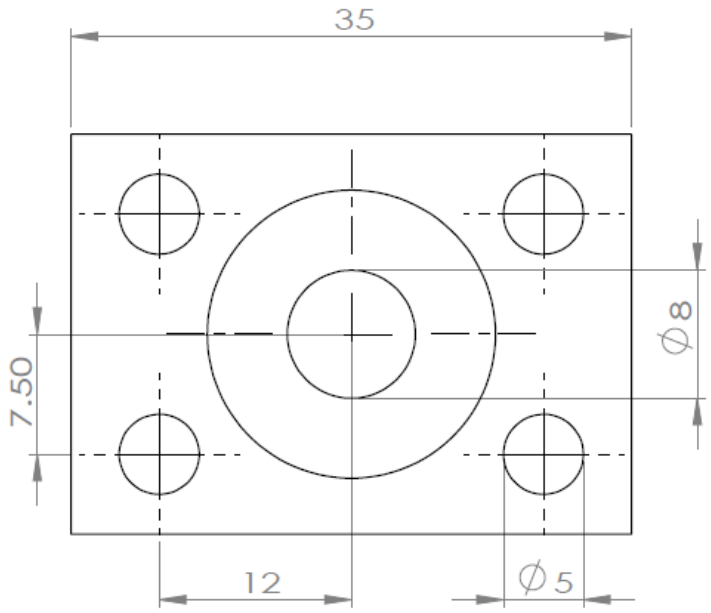
MATERIAL: STEEL ST 37			NUMBER : 1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 2:1	DWG NAME : MOTOR BASE	PART NUMBER : 5	

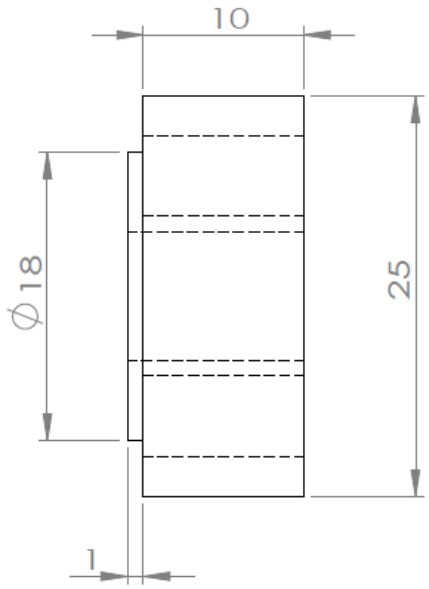




Unit : mm.

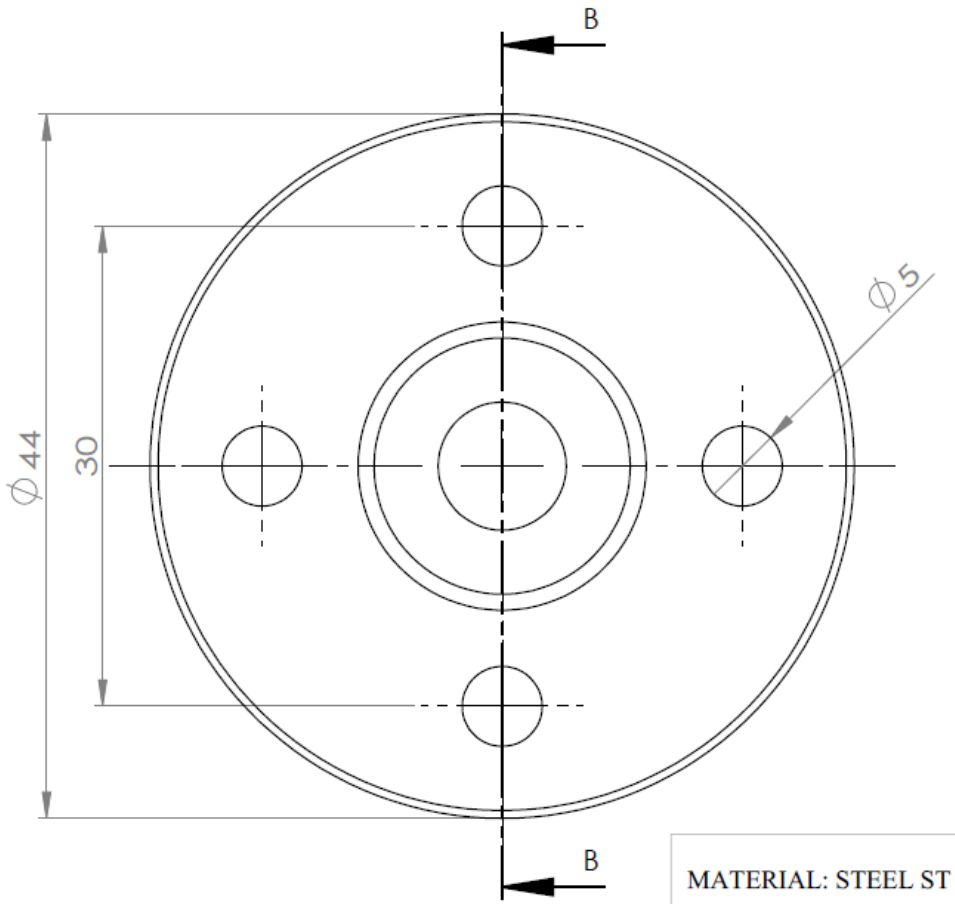
MATERIAL: STEEL ST 37			NUMBER :1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 1:2	DWG NAME : MOTOR COVER	PART NUMBER : 6	

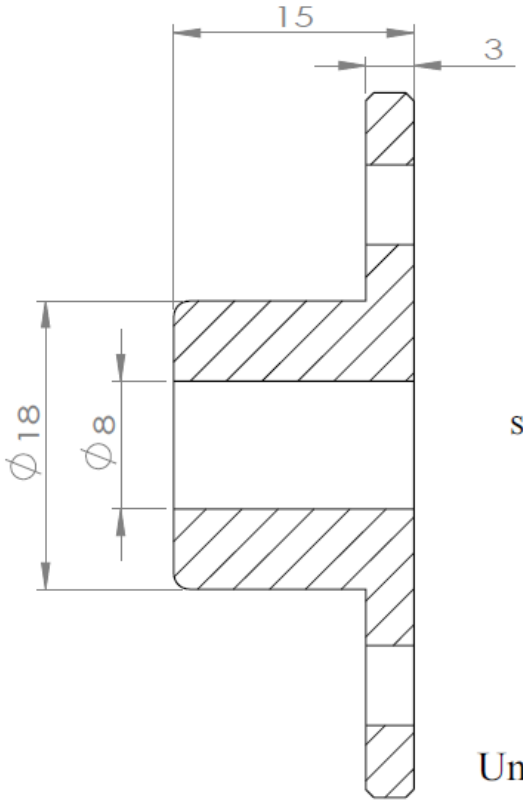




Unit : mm.

MATERIAL: ALUMNUM 6060			NUMBER :3
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 2:1	DWG NAME : BUSH BAERING	PART NUMBER : 7	

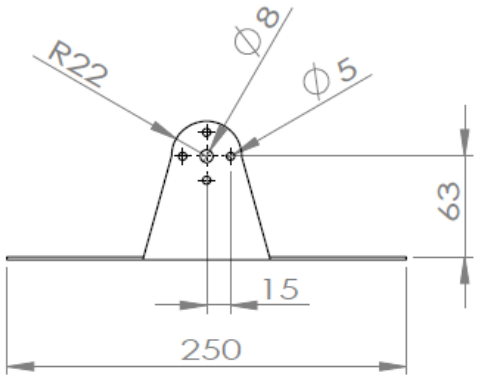


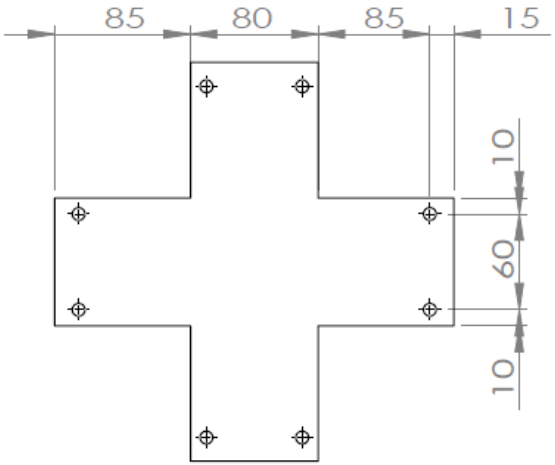


SECTION B-B

Unit : mm.

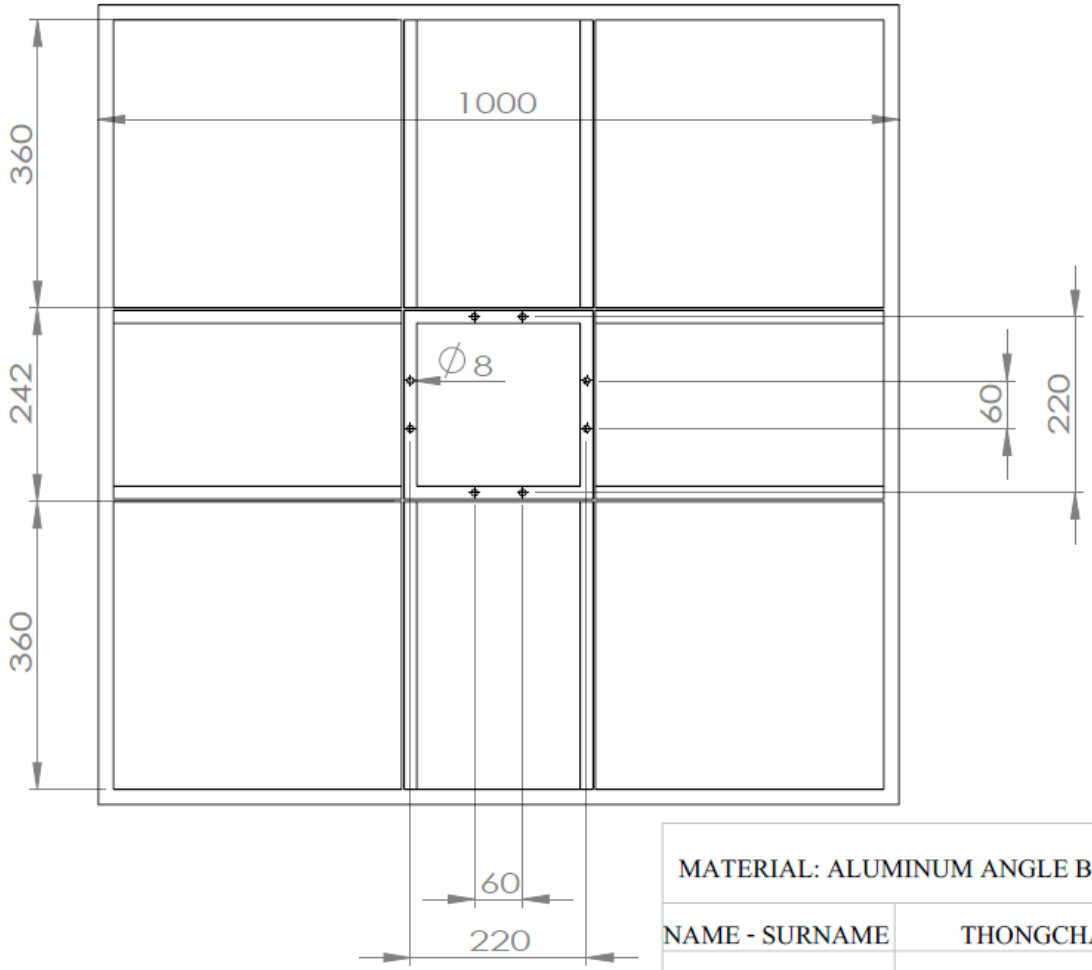
MATERIAL: STEEL ST 37			NUMBER :2
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 2:1	DWG NAME :BRACKET	PART NUMBER : 8	

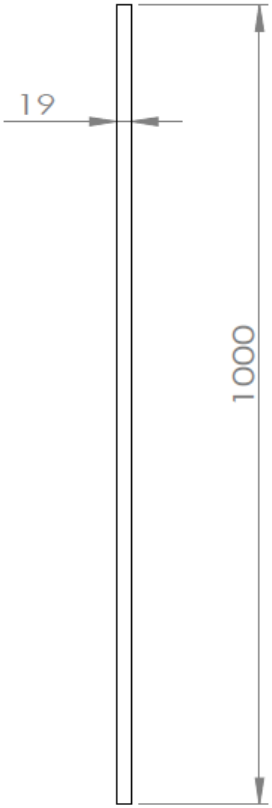




Unit : mm.

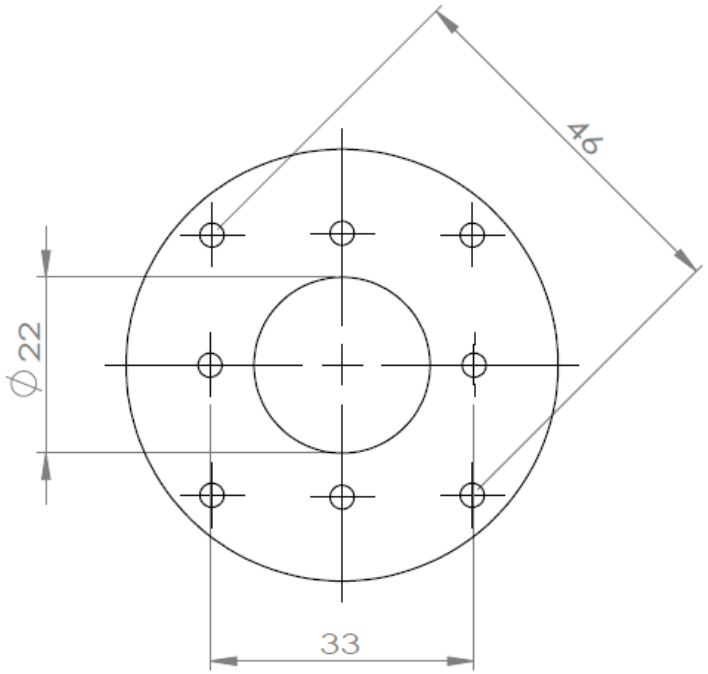
MATERIAL:STEEL ST 37			NUMBER : 1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 1:5	DWG NAME :PLATE	PART NUMBER : 9	

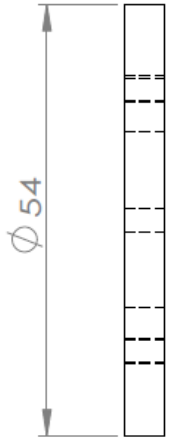




Unit : mm.

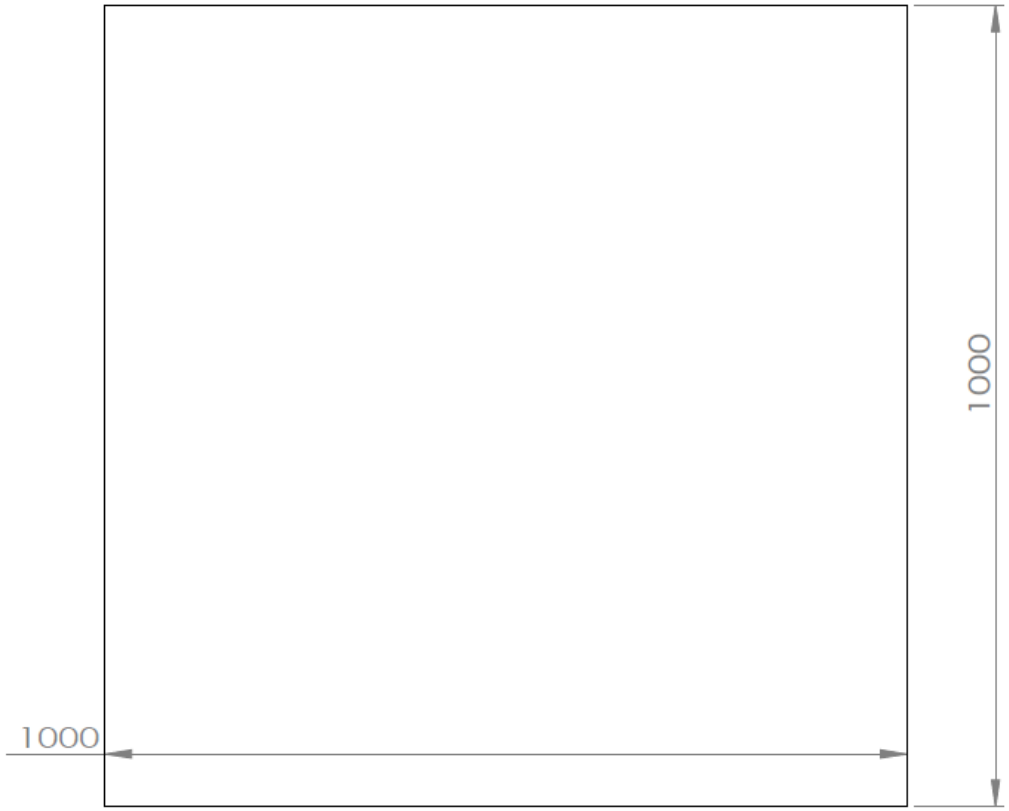
MATERIAL: ALUMINUM ANGLE BAR			NUMBER :1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 1:10	DWG NAME : MIRROR FRAME	PART NUMBER :10	

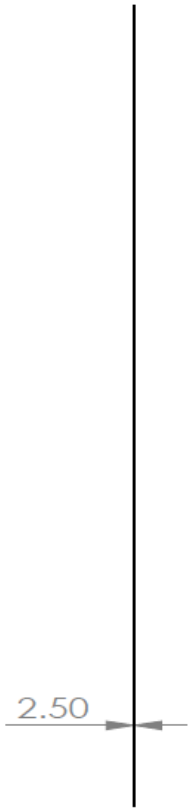




Unit : mm.

MATERIAL: ALUMINUM 6060			NUMBER :1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE : 1:1	DWG NAME : MOTOR SPACER	PART NUMBER : 11	





Unit : mm.

MATERIAL: ACRYLIC MIRROR			NUMBER :1
NAME - SURNAME	THONGCHAI PANNA	RAJAMANGALA UNIVERSITY OF TECHNOLOGY THANYABURI	
DEPARTMENT	MECHANICAL ENGINEERING		
GROUP	54343 MEE		
CODE	115430431026-5		
SCALE :1:5	DWG NAME : MIRROR	PART NUMBER : 12	