



รายงาน

เรื่อง การถ่ายภาพและส่งข้อมูลผ่านคลื่น FM กลุ่มที่ 1

ผู้จัดทำ

1. นายธนพล วงศ์อาชา	กลุ่มที่ 8	รหัสนักศึกษา 62010356
2. นายสุทธิราช ภูโภ	กลุ่มที่ 8	รหัสนักศึกษา 62010966
3. นายธนดล สินอนันต์วนิช	กลุ่มที่ 22	รหัสนักศึกษา 62010345
4. นายธนา ตึงประสม	กลุ่มที่ 22	รหัสนักศึกษา 62010381
5. นายธีรเดนย์ จันทร์หอม	กลุ่มที่ 44	รหัสนักศึกษา 62010436
6. นายสิริวิชญ์ สุขวัฒนาวิทย์	กลุ่มที่ 44	รหัสนักศึกษา 62010948

อาจารย์ที่ปรึกษา

รศ.ดร. อรฉัตร จิตต์ไสวภัตร์

ดร. จิรสักดิ์ สิทธิกร

รายงานนี้เป็นส่วนหนึ่งในรายวิชา Data Communications (01076007)

ภาคเรียนที่ 1 ปีการศึกษา 2563 ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คำนำ

รายงานนี้เป็นส่วนหนึ่งในรายวิชา Data Communications (01076007) โดยมีจุดประสงค์ เพื่อให้ได้ศักยภาพความรู้ในเรื่องการรับส่งข้อมูลผ่านคลื่น FM ได้อย่างเข้าใจและมีการปฏิบัติจริง โดยรายงานฉบับนี้มีเนื้อหาเกี่ยวกับการออกแบบการทำงานของระบบควบคุมกล้อง โดยมีการส่งข้อมูล ผ่านสัญญาณคลื่น FM โดยในงานนี้มีการใช้ความรู้ในเรื่องของการส่งข้อมูลแบบ Stop and Wait ARQ, การแปลงคลื่นสัญญาณ (Digital Modulation) แบบ FSK (Frequency Shift Keying), การ ออกแบบ Frame ในการส่งข้อมูล (Flow Control) และการควบคุมความผิดพลาดของข้อมูล (Error Control)

คณะผู้จัดทำหวังว่ารายงานเล่มนี้จะเป็นประโยชน์แก่ผู้อ่าน นักเรียนหรือนักศึกษาที่กำลัง ศึกษาเรื่องนี้อยู่ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ทางผู้จัดทำขอน้อมรับไว้และขอภัย มา ณ ที่นี่

คณะผู้จัดทำ

สารบัญ

เรื่อง	หน้า
คำนำ	ก
สารบัญ	ข
บทที่ 1 บทนำ	1
ที่มาและความสำคัญ	1
วัตถุประสงค์	1
ขอบเขตของโครงการ	1
บทที่ 2 เอกสารที่เกี่ยวข้อง	4
การเข้ารหัสสัญญาณ (Signal Modulation)	4
การมอดูเลตสัญญาณดิจิตอล	4
- Bit Rate and Baud Rate	5
- Carrier Signal	5
การมอดูเลตทางความถี่ (Frequency-Shift Keying : FSK)	6
การควบคุมอัตราการไหลของข้อมูล (Flow Control)	7
การควบคุมความผิดพลาดของข้อมูล (Error Control)	7
กลไกในการควบคุมอัตราการไหลและควบคุมความผิดพลาดของข้อมูล	8
- Stop-and-Wait ARQ	8
Machine Learning (ML) for detect images	9
- Convolutional Neural Network (CNN)	9
- Feature Extraction	10
- Max Pooling	10
การติดตั้งกล้อง OV7670 Camera Module	12
การสร้าง Server ด้วย python flask	16
ปัญหาครอสทอล์ค (Crosstalk)	17
บทที่ 3 ขั้นตอนและวิธีการดำเนินงาน	18
ขั้นตอนการดำเนินงาน	18
ระยะเวลาดำเนินงาน	18
การออกแบบระบบ	18
วัสดุอุปกรณ์และโปรแกรมที่ใช้ (Hardware & Software)	44

บทที่ 4	ผลการดำเนินงาน	45
	ผลลัพธ์การประมวลผลภาพด้วย Machine Learning (ML)	54
บทที่ 5	สรุปผลการดำเนินงาน	58
	สรุปผลการดำเนินงาน	58
	ปัญหาอุปสรรคและแนวทางแก้ไข	58
บทที่ 6	สิ่งที่ทำได้เพิ่มเติมหลังการทดลอง	60
	บรรณานุกรม	61
	ภาคผนวก	63

บทที่ 1

บทนำ

ที่มาและความสำคัญ

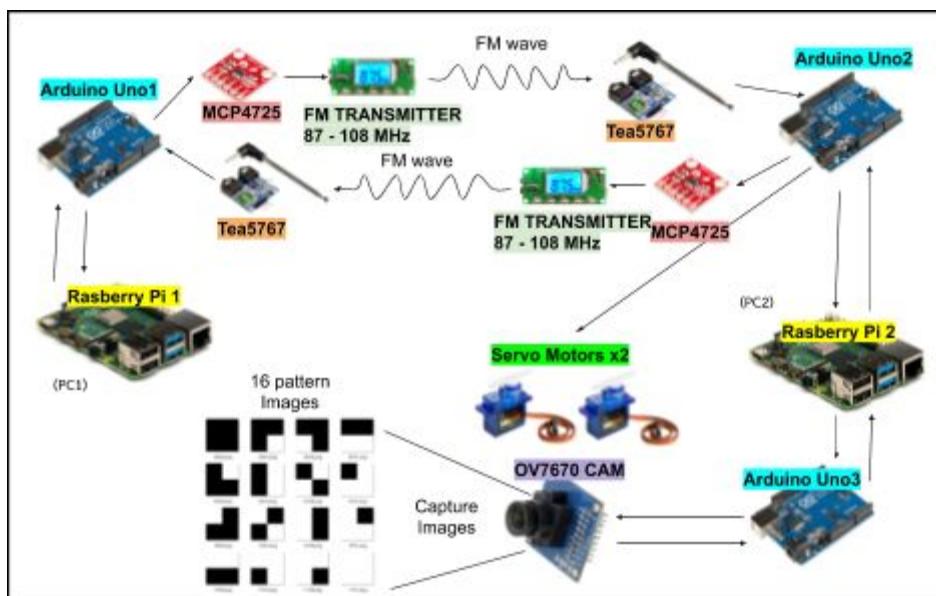
เนื่องด้วยการสื่อสารในปัจจุบันมีความสำคัญเป็นอย่างมาก ทางภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ได้มีการจัดการเรียนการสอนในรายวิชา Data Communications (01076007) จึงได้มีการให้นักศึกษาได้ฝึกการทำงานจริง ด้วยการทำโครงการถ่ายภาพและส่งข้อมูลผ่านคลื่น FM ซึ่งใช้ความรู้ทางด้านการสื่อสารทั้งการรับและส่งของข้อมูล การตรวจสอบความถูกต้องของข้อมูล การส่งสัญญาณผ่านคลื่นในรูปแบบต่าง ๆ รวมไปถึงการเข้ารหัสข้อมูลของผู้ส่งและการถอดรหัสข้อมูลของผู้รับ ทางคณะผู้จัดทำจึงมีแนวคิดที่จะนำความรู้ที่ได้ศึกษา นำมาประยุกต์ใช้และปฏิบัติจริง โดยการออกแบบระบบการทำงานรับส่งข้อมูลผ่านอุปกรณ์ Arduino ด้วยการใช้คลื่น FM ใน การส่งข้อมูล รูปแบบการส่งเป็นแบบ Stop and Wait ARQ พร้อมกับมีการตรวจสอบความผิดพลาดของข้อมูลด้วยวิธี CRC (Cyclic Redundancy Check)

วัตถุประสงค์

โครงการเรื่อง ถ่ายภาพส่งสัญญาณผ่านคลื่น FM มีวัตถุประสงค์ในการศึกษาค้นคว้า ดังนี้

- เพื่อศึกษาการทำงานส่งข้อมูลผ่าน Frequency Modulation (FM)
- ฝึกการออกแบบ Frame ในชั้น Data Link Layer (OSI Model)
- ฝึกการออกแบบ Flow control และ Error control ของเฟรมที่ส่ง

ขอบเขตของโครงการ



รูปที่ 1.1 แสดงภาพตัวอย่างการทำงาน

1. ส่วน Raspberry Pi ตัวที่ 1 (แทน PC1)

- สามารถสั่งให้ฝั่ง RasPi 2 เริ่มต้นการทำงาน (เก็บข้อมูลจากกล้องทุกมุมภาพ)
- แสดงผลสรุปข้อมูลรหัส Binary (0000, 0001, ..., 1111) และ มุม (-45, 0, +45) ที่ได้รับจากกล้องฝั่ง RasPi 2 มาแสดงผล
- หลังจากนั้น สามารถเลือก
 - สั่งให้ RasPi 2 เก็บค่าข้อมูลภาพตามรหัส Binary ที่ได้
 - RasPi 2 หมุนกล้องไปยังภาพตามรหัส Binary ที่สั่ง
 - RasPi 2 ส่งข้อมูลจุดภาพ (ระดับสีภาพ) ตามที่กล้องหมุนไป จำนวน 20 ค่า มาแสดงผลที่ RasPi 1
 - จำนวนข้อมูล 20 ค่า มาจาก Quadrant ละ 5 ค่า ประกอบด้วย
 - ค่าระดับสีภาพจำนวน 4 จุด
 - ค่าพิกัดแต่ละจุดค่าเฉลี่ยในแต่ละ Quadrant
 - สั่งให้เริ่มต้นทำงานใหม่

2. ส่วน Raspberry Pi ตัวที่ 2 (แทน PC2)

- รอรับคำสั่งเริ่มการทำงานจาก RasPi 1 (เก็บข้อมูลจากกล้องทุกมุม พร้อมวิเคราะห์ชนิดข้อมูลรหัส Binary ภาพ)
- ขั้นตอนการทำงานของ RasPi 2 วิเคราะห์ข้อมูลชนิดข้อมูลรหัส Binary ภาพ จากกล้องในมุมต่าง ๆ แล้วส่งข้อมูลสรุปให้ RasPi 1
- รอรับคำสั่งจาก RasPi 1
 - หาก RasPi 1 สั่งให้เก็บค่าข้อมูลภาพตามรหัส Binary
 - หมุนกล้องไปมุมภาพรหัส Binary ที่กำหนดจาก RasPi 1
 - เก็บข้อมูลภาพ
 - ส่งข้อมูลจำนวนจุดภาพตามที่กล้องหมุนไปกลับมา RasPi 1
 - หาก RasPi 1 สั่งให้เริ่มต้นทำงานใหม่ จึงเริ่มทำงานใหม่ตั้งแต่ต้น

3. ส่วน Tx, Rx, Camera

- ใช้เป็น Arduino

4. ส่วนการเชื่อมต่อระหว่าง Arduino <-> Raspberry Pi

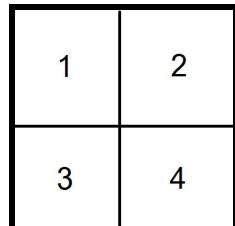
- เป็นสาย USB 2.0 Type B

5. Communication (ระหว่าง Raspberry Pi ตัวที่ 1 - Raspberry Pi ตัวที่ 2)

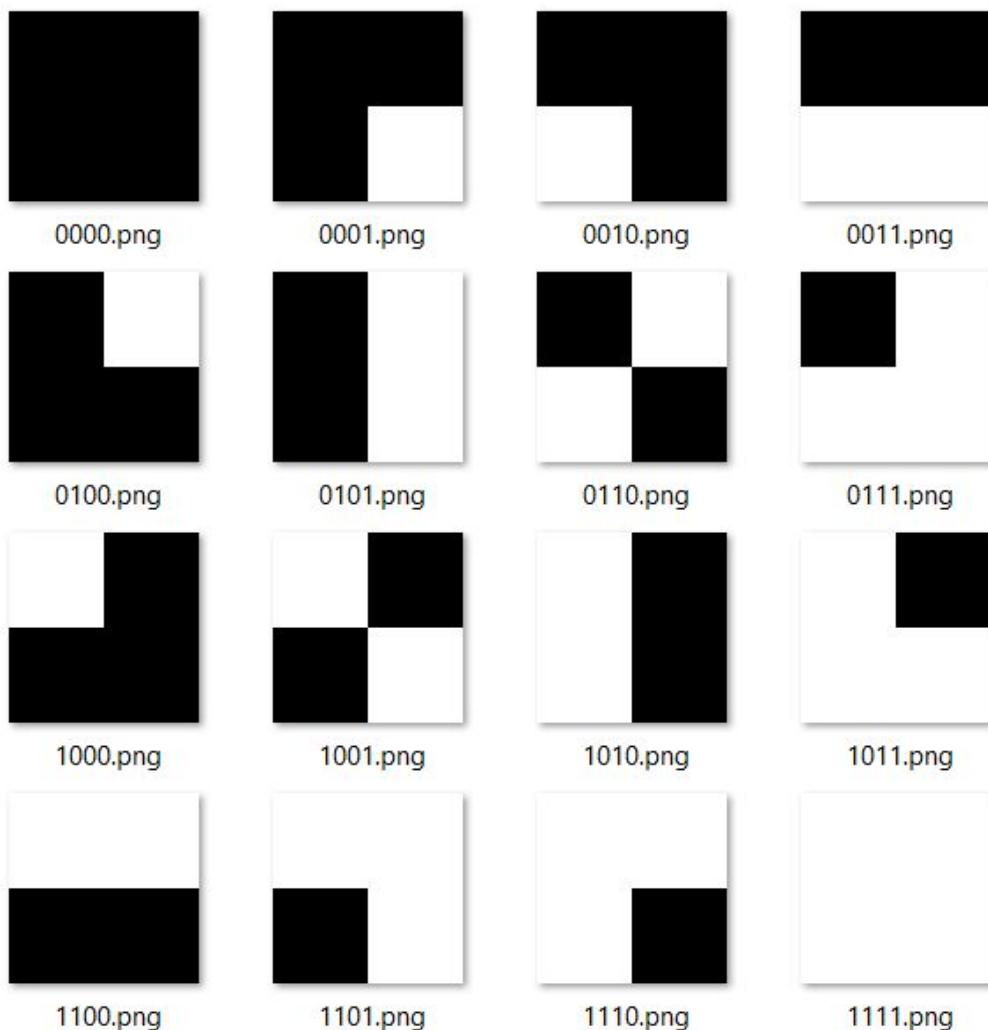
- ต้องใช้ Digital Modulation ที่เหมาะสม
- ส่งแบบ Wireless (FM)
- มี Error Detection
- มีการกำหนด Frame Design ที่เหมาะสม
- มีการกำหนด Flow & Error Control

6. ตัวอย่างภาพทั้งหมด 16 รูปแบบ (ตามรหัส Binary)

- ขนาดกระดาษภาพ 10x10 cm จำนวน 16 แผ่น (รูปแบบรหัส Binary)
 - สีดำ แทนเป็น 0
 - สีขาว แทนเป็น 1



รูปที่ 1.2 แสดงการกำหนดตำแหน่งตัวเลข



รูปที่ 1.3 ภาพทั้งหมด 16 รูปแบบตามรหัส Binary

บทที่ 2

เอกสารที่เกี่ยวข้อง

การเข้ารหัสสัญญาณ (Signal Modulation)

การmodulate (Modulation) จะเป็นหลักการที่สำคัญที่ใช้ในการสร้างสัญญาณอนาล็อก ก่อนที่จะถูกส่งออกไป ซึ่งมอดูลเตชันจะแบ่งออกเป็น 2 ประเภทคือ การmodulateสัญญาณดิจิตอล เช่น ASK (amplitude shift keying), FSK (frequency shift keying), PSK (phase shift keying) และ การmodulateตอนนาล็อก เช่น AM (amplitude modulation), FM (frequency modulation), PM (phase modulation) นอกจากนั้นแล้วจะได้พูดถึงเรื่องของโมเด็ม ซึ่งเป็นอุปกรณ์ที่ใช้ในการส่งสัญญาณอนาล็อกผ่านสายโทรศัพท์

การmodulateสัญญาณดิจิตอล

การแปลงบิตข้อมูลเป็นสัญญาณอนาล็อก หรือเรียกอีกอย่างว่า digital-to-analog modulation ตัวอย่างเช่น ถ้าเราต้องการที่จะส่งข้อมูลจากคอมพิวเตอร์เครื่องหนึ่งไปยังอีกเครื่องหนึ่งโดยใช้สายโทรศัพท์เป็นสื่อกลาง ข้อมูลที่เก็บอยู่ในคอมพิวเตอร์จะอยู่ในรูปแบบของดิจิตอล แต่สายโทรศัพท์จะต้องใช้สัญญาณอะนาล็อก ดังนั้นจะต้องทำการแปลงบิตข้อมูลให้เป็นสัญญาณอะนาล็อกเสียก่อน จึงจะสามารถที่จะส่งสัญญาณนั้นผ่านสายโทรศัพท์ได้

คุณลักษณะของคลื่นรูป Sine จะประกอบไปด้วย แอมเพลจูด, ความถี่ และเฟส ถ้าคุณลักษณะอย่างใดอย่างหนึ่งมีการเปลี่ยนแปลง นั่นหมายความว่าจะได้คลื่นรูป Sine คลื่นใหม่ที่ได้มีการเปลี่ยนแปลงไปจากเดิม จากหลักการนี้จึงสามารถนำมาประยุกต์ใช้กับสัญญาณดิจิตอลได้ เช่น การเปลี่ยนแปลงคุณลักษณะของคลื่นรูป Sine แต่ละครั้งจะหมายถึงบิตข้อมูลที่ได้มีการเปลี่ยนแปลงไป ดังนั้นในการแปลงบิตข้อมูลเป็นสัญญาณอะนาล็อกนั้น จะหมายถึงการเปลี่ยนคุณลักษณะของคลื่นรูป Sine นั่นเอง หรือเรียกอีกอย่างว่า การmodulateสัญญาณดิจิตอล ซึ่งสามารถแบ่งออกได้ดังนี้ Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) และ Phase Shift Keying (PSK) นอกจากนั้นแล้วยังมีอีกเทคนิคหนึ่งซึ่งมีประสิทธิภาพการทำงานที่ดีขึ้น โดยจะรวมวิธีการของ ASK และ PSK เข้าด้วยกัน เรียกเทคนิคแบบนี้ว่า Quadrature Amplituded Modulation (QAM)

ก่อนที่จะอธิบายถึงการmodulateสัญญาณดิจิตอลได้นั้น เราจะต้องทราบความหมายของศัพท์เทคนิคที่สำคัญ ๆ ก่อน ซึ่งได้แก่ bit rate, baud rate และ carrier signal

Bit Rate and Baud Rate

ในการมอดูลे�ตสัญญาณดิจิตอลนั้น มีคำที่ใช้งานกันอยู่บ่อย ๆ คือ อัตราบิต และอัตราบอต

- **อัตราบิต (Bit Rate)** หมายถึง จำนวนของบิตข้อมูลที่สามารถส่งได้ใน 1 วินาที มีหน่วยเป็น bit per second (bps)
- **อัตราบอต (Baud Rate)** หมายถึง อัตราของการเปลี่ยนแปลงสัญญาณใน 1 วินาที ใน การเปลี่ยนแปลงสัญญาณในแต่ละครั้งนั้นสามารถส่งข้อมูลหนึ่งบิตหรือมากกว่านั้น ก็ได้ เช่น ถ้าใน 1 วินาที สัญญาณมีการเปลี่ยนเฟส 5 ครั้ง อัตราบอตของสัญญาณนี้ จะเท่ากับ 5 บอตต่อวินาที (baud/s)

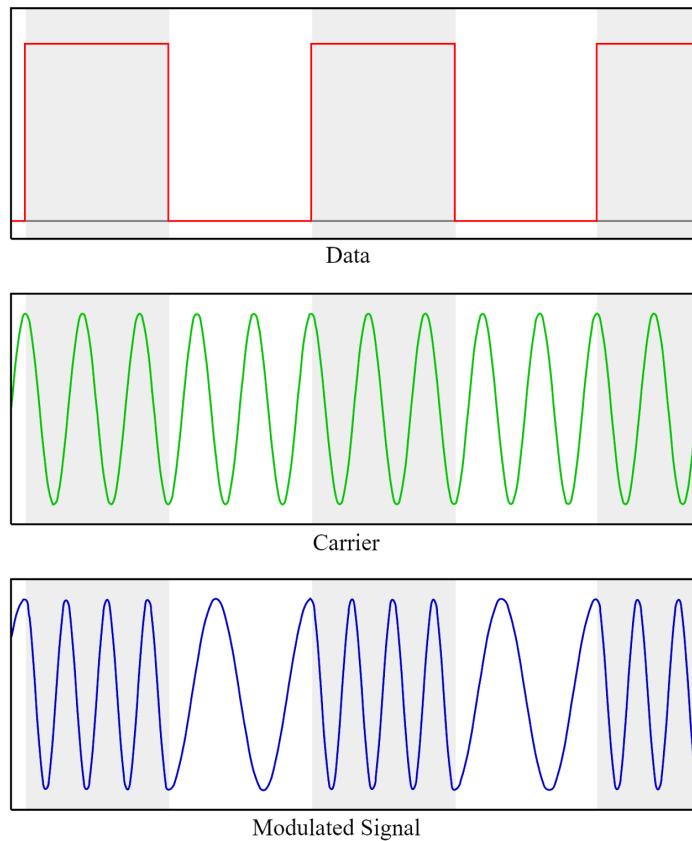
อัตราบอต จะมีค่าเท่ากับหรือน้อยกว่า อัตราบิต ถ้าจะเปรียบเทียบระหว่างอัตราบิต อัตราบอต กับการเดินทางด้วยรถยนต์แล้ว อัตราบอตจะเปรียบเสมือนรถยนต์ ส่วนอัตราบิตจะเปรียบ เสมือนผู้โดยสาร ถ้าสมมติให้รถยนต์ 1,000 คน และแต่ละคนไม่มีผู้โดยสารคนอื่น นอกจากคนขับ ต้องการเดินทางจากที่หนึ่งไปยังอีกที่หนึ่ง ซึ่งการเดินทางครั้งนี้จะมีคนที่สามารถเดินทางได้ทั้งหมด 1,000 คน แต่ถ้ารถยนต์แต่ละคัน มีผู้โดยสารเพิ่มอีกคันละ 3 คน (รวมคนขับด้วยเป็น 4) การเดินทางครั้งนี้จะมีคนที่สามารถเดินทางได้ทั้งหมด 4,000 คน จากตัวอย่างที่ได้กล่าวไปนั้นจะเห็นได้ว่า ปัญหาของการจราจรจะขึ้นอยู่กับจำนวนของรถบนท้องถนน ไม่ได้ขึ้นอยู่กับจำนวนของคนโดยสาร เช่นเดียวกันกับการส่งสัญญาณผ่านสื่อกลาง ซึ่งค่าที่เป็นตัวบ่งบอกถึงแบบดิจิตที่ต้องการนั้นคือ อัตราบอต ไม่ใช้อัตราบิต

Carrier Signal

สัญญาณคลื่นพาห์ (Carrier signal) เป็นสัญญาณที่ถูกสร้างขึ้นมาเพื่อที่จะใช้ในการรวมกับ สัญญาณของข้อมูล เนื่องจากสัญญาณของข้อมูลอาจจะมีความถี่ไม่เหมาะสมกับช่องสัญญาณ ดัง นั้นอุปกรณ์ที่จะทำการส่งสัญญาโนาล็อกนั้นจะต้องสร้างสัญญาณคลื่นพาห์ขึ้นมาก่อน ซึ่งจะมี ความถี่ที่เหมาะสมกับช่องสัญญาณนั้น ๆ จากนั้นจึงนำสัญญาณคลื่นพาห์นี้รวมกับสัญญาณของ ข้อมูล ซึ่งจะทำให้สัญญาณคลื่นพาห์มีคุณลักษณะที่เปลี่ยนไปจากเดิม เช่น ค่าแอมพลิจูด ความถี่ หรือเฟส เป็นต้น กระบวนการรวมสัญญาณคลื่นพาห์กับสัญญาณของข้อมูล จะเรียกว่า มอดูลेशัน (Modulation of shift keying) ส่วนสัญญาณของข้อมูลเราสามารถเรียกอีกอย่างได้ว่า Modulating Signal

การมอดูลे�ตทางความถี่ (Frequency-Shift Keying : FSK)

เทคนิคการมอดูลे�ตข้อมูลกับสัญญาณคลื่นพาห์แบบ FSK นั้นจะทำให้ความถี่เปลี่ยนแปลงไปตามบิตของข้อมูล โดยให้แอมเพลจูด และเฟสคงที่ เช่น ถ้าบิตมีค่า ‘1’ จะให้ความถี่มีค่าสูงกว่าปกติ และบิตมีค่า ‘0’ จะให้ความถี่มีค่าน้อยกว่าปกติ โดยการส่งสัญญาณแบบ FSK จะสามารถแทนต่อสัญญาณรบกวนได้ดีกว่า ASK เนื่องจากอุปกรณ์ที่รับข้อมูลไม่ต้องสนใจว่าแอมเพลจูด หรือแรงดันไฟฟ้าจะมีค่าที่เปลี่ยนแปลงไปอย่างไร จะพิจารณาเฉพาะความถี่เท่านั้น แต่ FSK จะมีข้อเสียที่สืบที่ใช้ในการส่งสัญญาจนั้นจะต้องมีแบบดิจิตอลที่เพียงพอที่จะสามารถส่งผ่านความถี่ของสัญญาณที่ต้องการได้



รูปที่ 2.1 แสดงภาพสัญญาณการส่งข้อมูลแบบ FSK

การควบคุมอัตราการไหลของข้อมูล (Flow Control)

การควบคุมอัตราการไหลของข้อมูล เป็นการกล่าวถึงขั้นตอนกระบวนการที่ควบคุมจำนวนของข้อมูลที่ส่งออกไปให้อยู่ในปริมาณที่เหมาะสม ก่อนที่จะได้รับการตอบรับยืนยันจากผู้รับข้อมูล นั่นหมายความว่าข้อมูลที่ถูกส่งไปนั้นจะต้องถูกจำกัดเอาไว้จำนวนหนึ่ง ถ้ายังไม่ได้รับการตอบรับจากผู้รับว่าได้ข้อมูลชุดนั้น ๆ แล้ว ผู้ส่งจะต้องไม่ส่งข้อมูลชุดต่อไป สาเหตุที่ต้องมีการควบคุมอัตราการไหลก็เนื่องจากว่าผู้รับอาจมีความเร็วในการรับข้อมูลไม่เท่ากับผู้ส่ง หรือมีหน่วยความจำในการเก็บข้อมูลอย่างจำกัด หรือมีความเร็วในการประมวลผลต่ำ ดังนั้นการควบคุมอัตราการไหลของข้อมูลจึงมีความจำเป็นอย่างมาก ถ้าผู้รับยังไม่พร้อมจะรับข้อมูลในขณะนั้น ผู้ส่งจะต้องทำการหยุดส่งข้อมูลชั่วคราวเสียก่อน หรือถ้าผู้รับมีความเร็วในการประมวลผลต่ำ ผู้ส่งจะต้องส่งข้อมูลด้วยอัตราเร็วที่เหมาะสม เพื่อให้เกิดความสมดุลในการรับและส่งข้อมูล

การควบคุมความผิดพลาดของข้อมูล (Error Control)

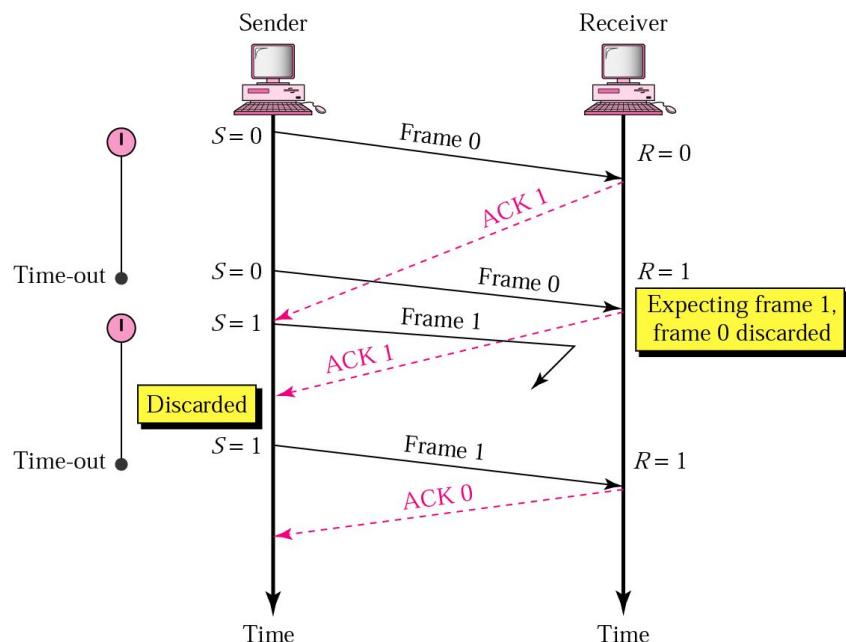
ในเดาลิงก์เลเยอร์นั้น การควบคุมความผิดพลาดของข้อมูลจะหมายถึง การที่ผู้ส่งต้องส่งข้อมูลไปใหม่อีกครั้งหนึ่ง ถ้าผู้รับไม่สามารถรับข้อมูลหรือได้รับข้อมูลที่ไม่ถูกต้อง สาเหตุที่ต้องมีการควบคุมเนื่องจากว่าข้อมูลจะต้องเดินทางจากที่หนึ่งไปยังอีกที่หนึ่ง จึงมีความเป็นไปได้ที่ข้อมูลชุดนั้นจะเกิดสูญหายหรือเสียหายในระหว่างการเดินทางได้ ดังนั้นผู้รับจะต้องมีกระบวนการในการตรวจสอบความผิดพลาดของข้อมูลนognานี้จะต้องมีการส่งข้อความไปบอกกับผู้ส่งข้อมูลด้วยว่าข้อมูลชุดนั้น ๆ ไปใหม่อีกครั้งหนึ่ง

กลไกในการควบคุมอัตราการไหลและความคงความผิดพลาดของข้อมูล

Stop-and-Wait ARQ

เป็นกลไกที่ง่ายและไม่มีความซับซ้อนมากนัก มีหลักการดังต่อไปนี้

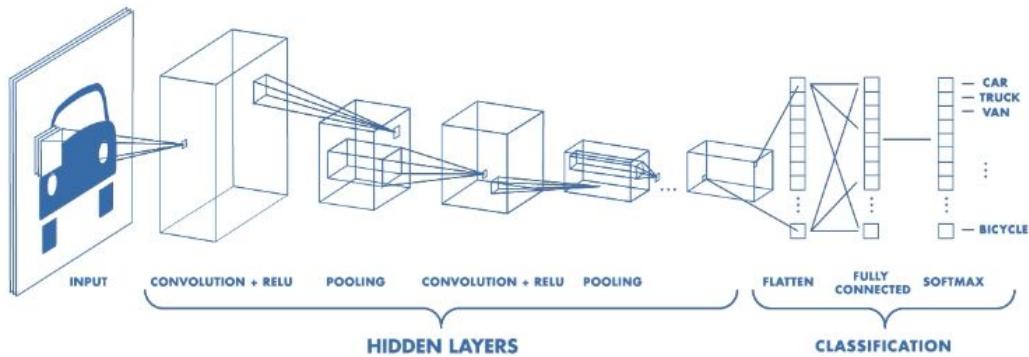
- ผู้ส่งข้อมูลจะต้องทำการสำเนาเฟรมข้อมูลที่จัดส่งไปเอาไว้ก่อนจนกว่าผู้ใช้จะยืนยันว่าได้รับเฟรมข้อมูลนั้นแล้ว เนื่องจากว่าถ้าเฟรมข้อมูลนั้นเกิดการสูญหายหรือเสียในระหว่างการส่ง ผู้ส่งจะสามารถส่งเฟรมข้อมูลใหม่ได้
- ในกรณียืนยันตอบรับเฟรมข้อมูลของผู้รับนั้น ผู้รับจะต้องส่งเฟรม acknowledgment (ACK) มาให้กับผู้ส่ง ซึ่งการส่งเฟรม ACK จะเป็นการบ่งบอกว่าได้รับเฟรมอะไร และเฟรมข้อมูลที่ต้องการถัดไปนั้นคือเฟรมอะไร
- ถ้าผู้รับข้อมูลได้รับเฟรมข้อมูลที่มีความผิดพลาด ผู้รับจะทำการหักเฟรมนั้นไป หรือถ้าได้รับเฟรมที่ไม่ต้องการ ก็จะทำการหักเฟรมนั้นไป เช่นกัน
- ผู้ส่งจะใช้ตัวแปร S ในการเก็บข้อมูลว่าได้ส่งเฟรมอะไรออกไป ส่วนผู้รับจะใช้ตัวแปร R ในการเก็บข้อมูลว่าเฟรมถัดไปที่ต้องการคือเฟรมอะไร
- ผู้ส่งจะมีการกำหนดเวลาเอาไว้หลังจากส่งเฟรมข้อมูลออกไปแล้ว ถ้าไม่ได้รับเฟรม ACK กลับมาในเวลาที่กำหนด จะต้องทำการส่งเฟรมข้อมูลนั้นกลับไปอีกครั้งหนึ่ง
- ผู้รับจะส่งเฟรม ACK กลับไปเมื่อได้รับเฟรมข้อมูลที่ไม่มีความผิดพลาดและเป็นเฟรมที่ต้องการ ถ้าผู้รับได้รับเฟรมที่ผิดพลาดหรือเฟรมที่ไม่ต้องการ จะไม่มีการส่งเฟรมใดๆ ตอบกลับไป



รูปที่ 2.2 การสื่อสารของโปรโตคอล Stop-and-Wait

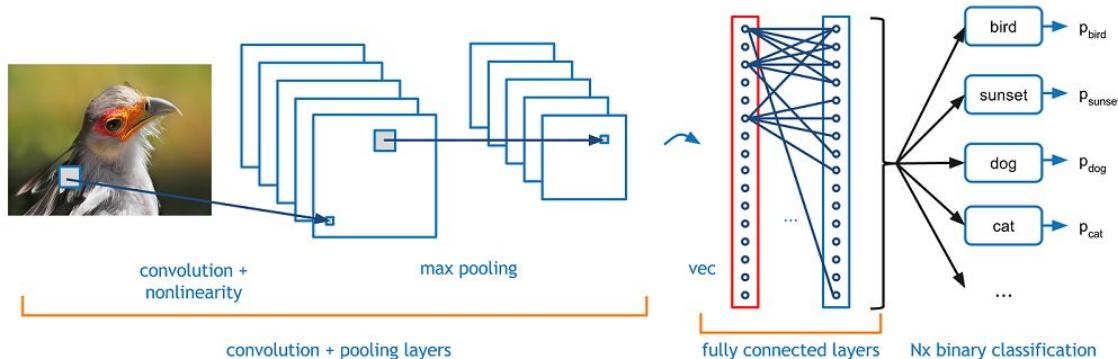
Machine Learning (ML) for detect images

Convolutional Neural Network (CNN)



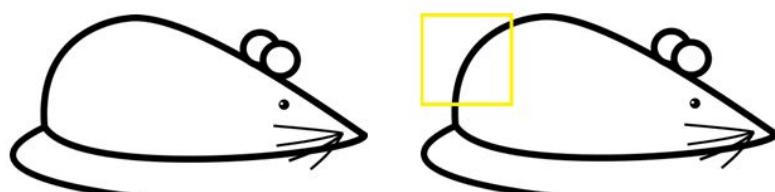
รูปที่ 2.3 แสดงภาพจำลองของ CNN

Convolutional Neural Network (CNN) หรือ โครงข่ายประสาทแบบคอนโวลูชัน เป็นโครงข่ายประสาทเทียมหนึ่งในกลุ่ม bio-inspired โดยที่ CNN จะจำลองการมองเห็นของมนุษย์ที่มองพื้นที่เป็นที่ย่อยๆ และนำกลุ่มของพื้นที่ย่อยๆ มาผสานกัน เพื่อดูว่าสิ่งที่เห็นอยู่เป็นอะไรกันแน่



รูปที่ 2.4 แสดงแสดงภาพจำลองส่วนประกอบต่างๆของ CNN

การมองพื้นที่ย่อยของมนุษย์ จะมีการแยกคุณลักษณะ (feature) ของพื้นที่ย่อยนั้น เช่น ลายเส้นและการตัดกันของสีซึ่งการที่มนุษย์รู้ว่าพื้นที่ตรงนี้เป็นเส้นตรงหรือสีตัดกัน เพราะมนุษย์ดูทั้งจุดที่สนใจและบริเวณรอบ ๆ ประกอบกัน



รูปที่ 2.5 แสดงการจำลองจุด focus ของสายตามนุษย์

Feature Extraction

แนวคิดของ CNN นั้นค่อนข้างเป็นแนวคิดที่ดีมาก แต่สิ่งที่ซับซ้อนของมันคือระบบการคำนวณที่สอดคล้องกับ Concept ของมันเองและต้องมีคณิตศาสตร์มารองรับ โดยการคำนวณตามแนวคิดนี้ใช้หลักการเดียวกันกับ convolution โคนโกลุชันเชิงพื้นที่ (Spatial Convolution) ในการทำงานด้าน Image Processing

การคำนวณนี้จะเริ่มจากการกำหนดค่าใน ตัวกรอง (filter) หรือ เครื่องเนล (kernel) ที่ช่วยดึงคุณลักษณะที่ใช้ในการรู้จ้าวต่อจากโดยปกติตัวกรอง/เครื่องเนลอันหนึ่งจะดึงคุณลักษณะที่สนใจออกมายังหนึ่งอย่างเราจึงจำเป็นต้องตัวกรองหลายตัวกรองด้วย เพื่อหาคุณลักษณะทางพื้นที่หลายอย่างประกอบกัน

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

รูปที่ 2.6 แสดงการทำ Convolution filter

Max Pooling

เรามองลองดูหนึ่งในปัญหาของการทำ CNN กันก่อน สมมติเราใช้ CNN ด้วยขนาดตัวกรอง 3x3 พิกเซล แต่เรารู้ดีว่าเวลาเรามองภาพแล้วเราตอบได้ว่ามันคืออะไร เพราะเรามองไปในบริเวณที่กว้างกว่านั้น



รูปที่ 2.7 แสดงการทำ Max Pooling

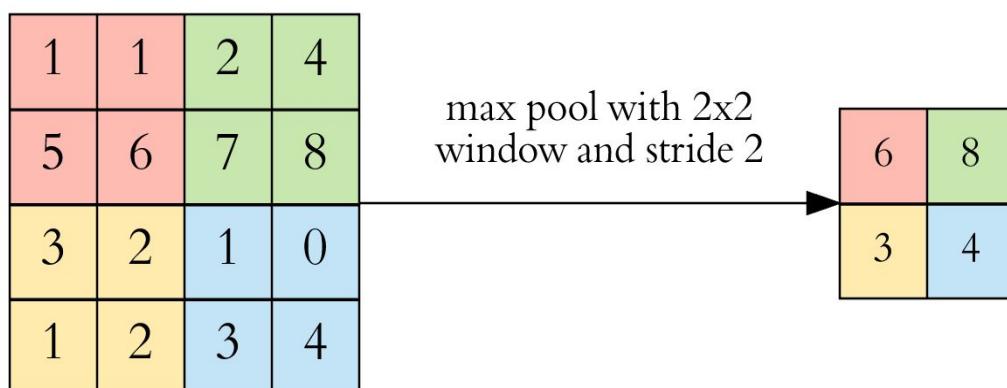
จากภาพ จะเห็นว่า ต่อให้รูปภาพมีขนาดสเกลที่เล็กลง แต่เรายังสามารถมองออกว่ามันคือเครื่องปั้นดินเผา แสดงว่าเราจำแนกวัตถุขึ้นนี้ที่ความละเอียดต่ำลง แต่เรากำลังทำ CNN ที่ความละเอียดสูง

Multiscale Analysis จากปัญหาด้านบนเราจะเห็นว่าเป็นไปได้ยากมากหากเราต้องอาศัยข้อมูลที่หยาบหรือละเอียดอย่างใดอย่างหนึ่งในการจำแนกวัตถุดังนั้นในการฝึกเครื่องเรางึงจะเป็นต้องมีข้อมูลทั้งหยาบและละเอียดควบคู่กันไป

ตอนนี้เรารู้แล้วว่าเราจำเป็นต้องคำนวณภาพในหลายสเกล แต่ปัญหาที่สำคัญคือเราจะทำให้การคำนวณอยู่ในรูปหลายสเกลได้อย่างไร หากเราใช้ตัวกรองขนาด 3×3 เราจะต้องจัดการกับรายละเอียดเล็กๆ (ภาพใหญ่มีรายละเอียดมาก จึงถือว่าเป็นสเกลละเอียด) แต่ด้วยตัวกรองขนาดเท่าเดิม หากทำกับภาพที่ขนาดเล็กลงแล้ว - มันจะครอบคลุมพื้นที่วัตถุเดิมมากขึ้น ดังนั้นถ้าโครงข่ายเรารู้จะต้องมีการย่อรูปประกอบด้วย เราจะสามารถเข้าถึงความสามารถด้านการวิเคราะห์หลายความละเอียดได้

Pooling คือความสามารถในการย่อรูปแบบหนึ่ง ซึ่งมีสองประเภทหลักที่นิยมกันคือ max pooling และ mean pooling

Max Pooling เป็นตัวกรองแบบหนึ่งที่หาค่าสูงสุดในบริเวณที่ตัวกรองทابอยู่มาเป็นผลลัพธ์โดยเราจะเตรียมตัวกรองในลักษณะเดียวกับการทำ Feature Extraction ของ CNN มาทางบนข้อมูลแล้วเลือกค่าที่สูงที่สุดบนตัวกรองนั้นมาเป็นผลลัพธ์ใหม่ และจะเลื่อนตัวกรองไปตาม Stride ที่กำหนดไว้ โดยขนาดตัวกรองของการทำ max pooling จะนิยมเรียกว่า pool size

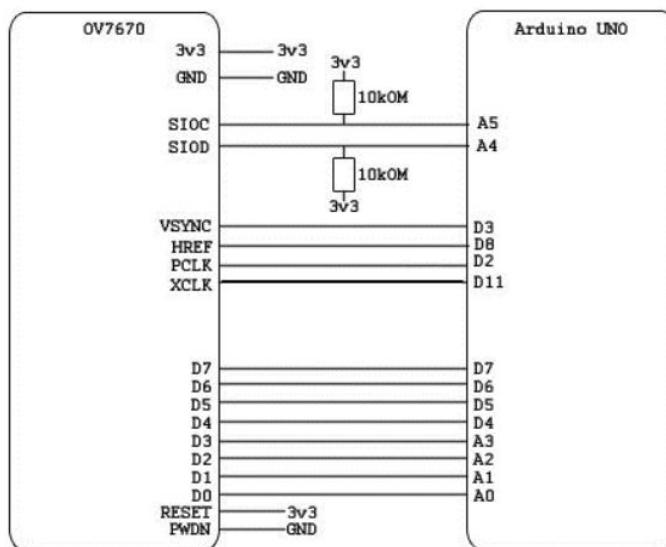


รูปที่ 2.8 แสดงการทำ max pool with 2×2

การติดตั้งกล้อง OV7670 Camera Module

ขั้นตอนการใช้งาน OV7670 (โมดูลกล้อง) ใช้สำหรับการทดสอบว่ากล้องสามารถทำงานได้หรือไม่

- ดาวน์โหลดไฟล์เดอร์ OV7670 DataCom จาก <https://drive.google.com/drive/folders/0B8uPgCHycl9VLXdvUWlxTi0ZlU?usp=sharing>
- ต่อโมดูลกล้องเข้ากับ Arduino UNO ดังภาพ โดย XCLK ไม่ต้องต่อตัวต้านทาน

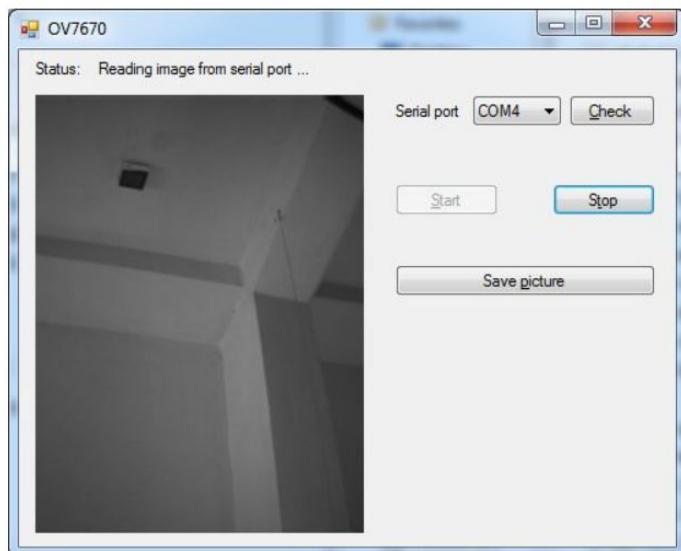


รูปที่ 2.9 การเชื่อมต่อโมดูลกล้อง OV7670 เข้ากับ Arduino UNO

Pin	Type	Description
VDD**	Supply	Power supply
GND	Supply	Ground level
SDIOC	Input	SCCB clock
SDIOD	Input/Output	SCCB data
VSYNC	Output	Vertical synchronization
HREF	Output	Horizontal synchronization
PCLK	Output	Pixel clock
XCLK	Input	System clock
D0-D7	Output	Video parallel output
RESET	Input	Reset (Active low)
PWDN	Input	Power down (Active high)

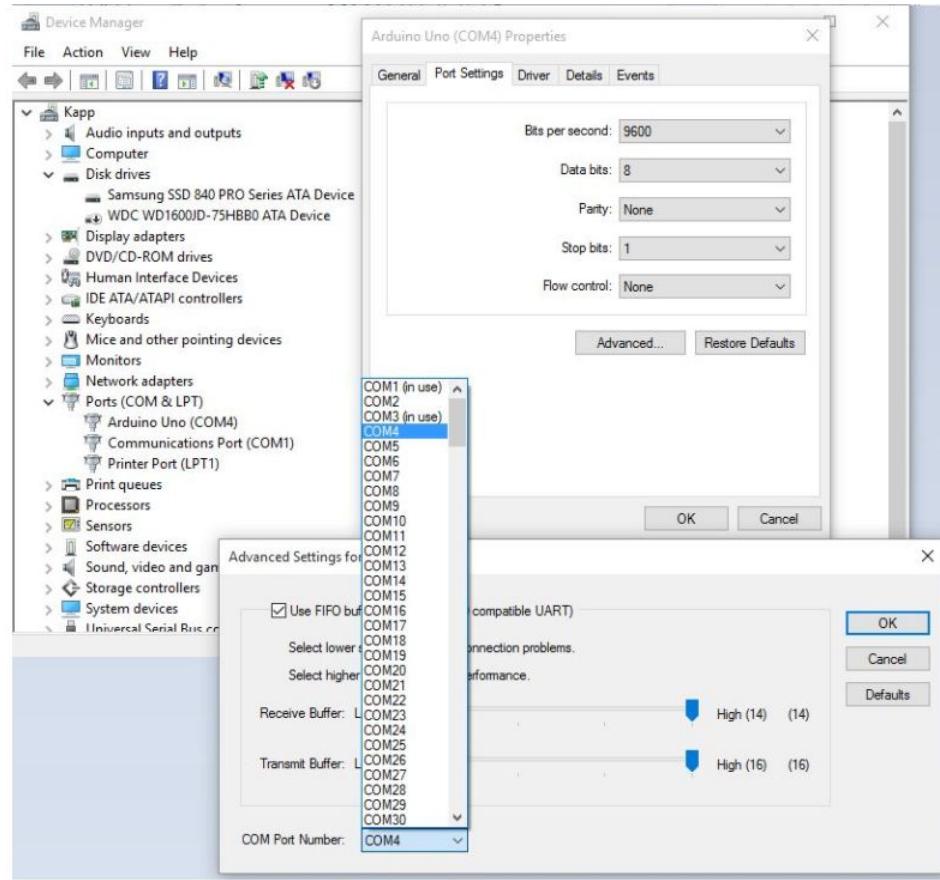
รูปที่ 2.10 คำอธิบายขาพินของโมดูลกล้อง OV7670

3. เปิดไฟลเดอร์ Camera ในไฟลเดอร์ OV7670 DataCom จากนั้นอัพโหลดโปรแกรม Camera.ino เข้าไปใน Arduino UNO
4. หากต่อทุกอย่างถูกต้องและ Upload ได้ถูกต้องสามารถดูภาพได้จากโปรแกรม ReadSerialPort.exe ได้ โดยเลือก comport ให้ถูกต้อง ให้กดปุ่ม start



รูปที่ 2.11 ตัวอย่างการรับภาพจากโมดูลกล้อง OV7670

5. ทำการเปลี่ยนเลข Comport โดยการเข้าไปที่ Device Manager, Ports, คลิกขวาที่ Arduino Uno, Properties, Port Setting, เปลี่ยน Port Number เป็น Com 1-5 (พอร์ตใดก็ได้) ดังภาพด้านล่าง



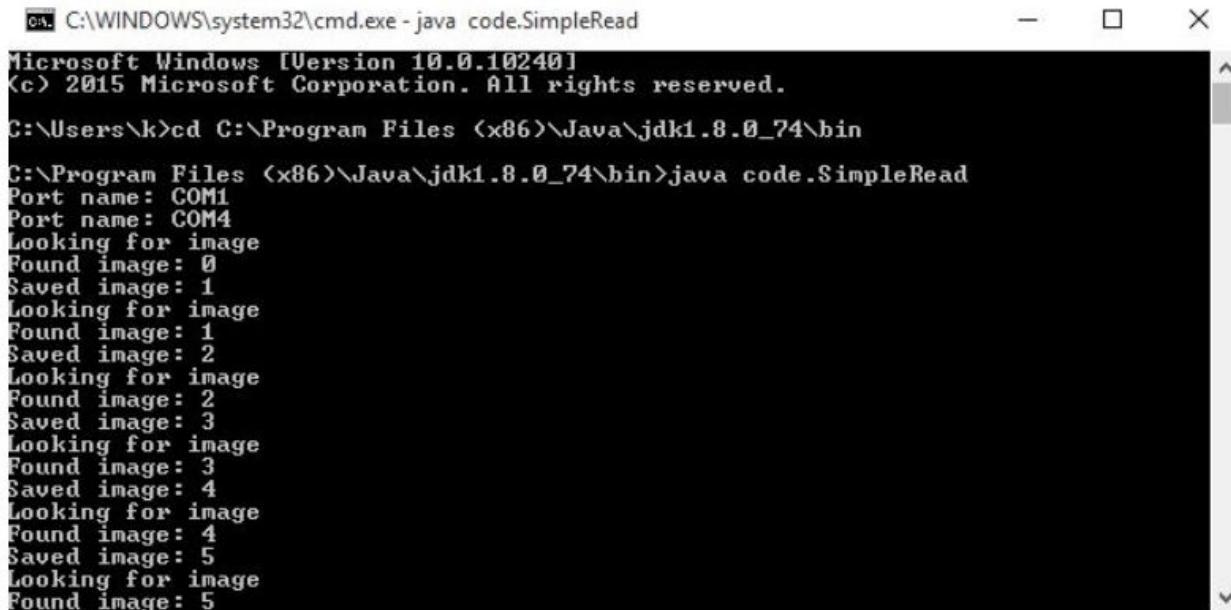
รูปที่ 2.12 การเปลี่ยนแปลงค่า Comport

6. สร้างโฟลเดอร์ชื่อ out ใน Drive C
7. ทำการติดตั้งโปรแกรม jdk-8u74-windows-i68 (มีให้ใน OV7670 DataCom)
8. Extract com_X.rar (โดย X คือเลข Comport ที่ได้ตั้งไว้ในขั้นตอนที่ 4)



รูปที่ 2.13 การ extract ไฟล์ที่ได้รับมา

9. Copy ไฟล์เดอร์ code ที่อยู่ใน com_X และวางใน C:\Program Files (x86)\Java\jdk1.8.0_74\bin
10. Copy file win32com.dll และวางใน C:\Program Files (x86)\Java\jdk1.8.0_74\jre\lib
11. Copy file comm.jar ในไฟล์เดอร์ lib และวางใน C:\Program Files (x86)\Java\jdk1.8.0_74\jre\lib\ext
12. Copy file javax.comm.properties ในไฟล์เดอร์ lib และวางใน C:\Program Files (x86)\Java\jdk1.8.0_74\jre\lib
13. เปิด cmd พิมพ์ cd C:\Program Files (x86)\Java\jdk1.8.0_74\bin
14. พิมพ์ java code.SimpleRead
15. หากทุกอย่างถูกต้อง cmd จะขึ้นข้อความดังภาพ และสามารถรูปภาพใน C:\out



```
C:\WINDOWS\system32\cmd.exe - java code.SimpleRead
Microsoft Windows [Version 10.0.10240]
© 2015 Microsoft Corporation. All rights reserved.

C:\Users\k>cd C:\Program Files (x86)\Java\jdk1.8.0_74\bin
C:\Program Files (x86)\Java\jdk1.8.0_74\bin>java code.SimpleRead
Port name: COM1
Port name: COM4
Looking for image
Found image: 0
Saved image: 1
Looking for image
Found image: 1
Saved image: 2
Looking for image
Found image: 2
Saved image: 3
Looking for image
Found image: 3
Saved image: 4
Looking for image
Found image: 4
Saved image: 5
Looking for image
Found image: 5
```

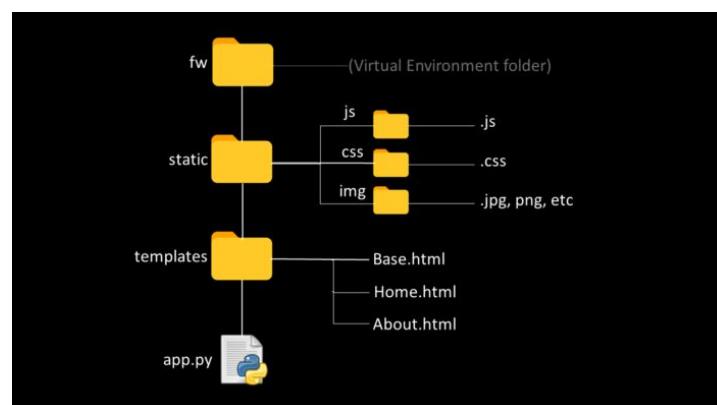
รูปที่ 2.14 ตัวอย่างการแสดงผลเมื่อตั้งค่าถูกต้องบน cmd

การสร้าง Server ด้วย python flask



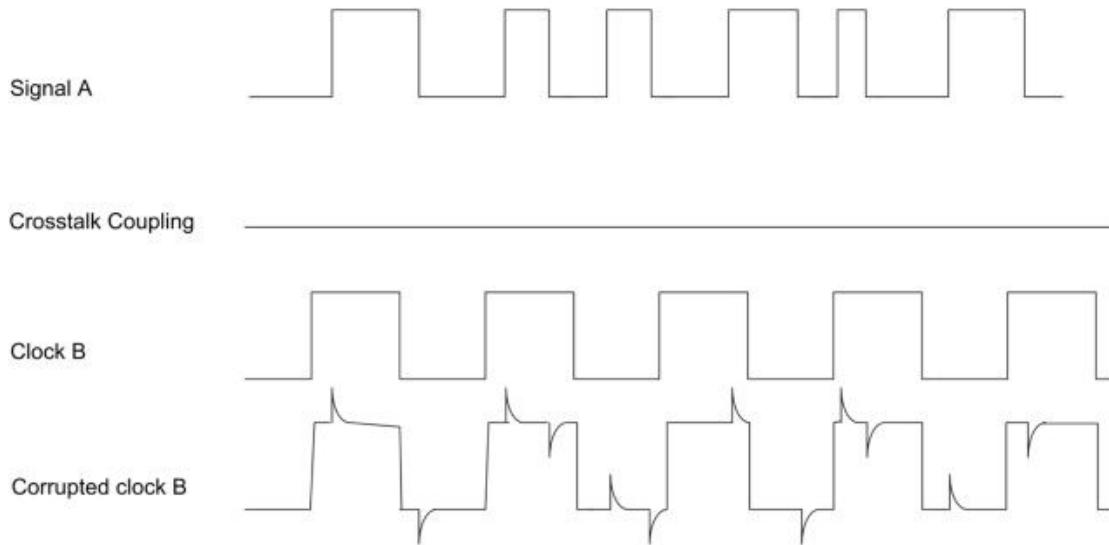
รูปที่ 2.15 Logo Flask

1. Flask คือ web framework ที่เขียนขึ้นมาสำหรับ Python เพื่อใช้ร่วมกัน webserver โดย flask ถูกเรียกว่า micro framework เพราะว่า เป็นระบบไม่ต้องการเครื่องมือ หรือ library อะไรเพิ่มเติม สามารถทำงานด้วยตัวเองได้
2. ศึกษาการ Stream ค่า โดยใช้ flask โดยนำมาใช้รับการ Stream ค่าจาก Serial port ไปยังหน้า Website แบบ Realtime
3. ศึกษาการสร้าง Route ไว้ใช้ในการ ส่งค่าจาก Web Client มาจาก Server โดยเป็นคำสั่ง ที่ต้องการส่งไปยัง Arduino
 - a. Route คือ class ที่ใช้จัดการกับเส้นทาง เข้า-ออก ของเว็บไซต์เรา พุดให้เข้าใจง่ายๆคือ เป็นตัวกำหนด Path ว่า Path นี้ให้ทำงานที่ Controller ไหน หรือ แสดง View ไหน รวมถึงสามารถรับการส่ง Parameter ผ่านเจ้าตัว Route ได้ด้วย
4. โครงสร้างของ Flask



รูปที่ 2.16 แสดงโครงสร้างของ Flask

ปัญหาคอสทอลค์ (Crosstalk)



รูปที่ 2.17 แสดงการเกิด Noise ในขา Corrupted clock B

การแทรกสัญญาณข้าม หรือ สัญญาณแทรกข้าม (อังกฤษ: crosstalk (XT)) หมายถึง สัญญาณไฟฟ้าข้ามไปปรบกวนกัน เป็นปรากฏการณ์ที่เกิดขึ้นได้ในการส่งสัญญาณในขณะที่สัญญาณเดินทางไปในสื่อสายทองแดงด้วยความเร็วสูง ปรากฏการณ์นี้มักพบได้เมื่อสื่อที่ใช้ในการส่งสัญญาณเป็นสายทองแดงตั้งแต่สองจุดขึ้นไป เช่น โทรศัพท์ ซึ่งมีสายทองแดงมัดรวมกันเป็นคู่ นอกจากนี้ ปรากฏการณ์นี้ยังสามารถเกิดจากสนามแม่เหล็กภายในสายทองแดงเองขณะกำลังส่งสัญญาณ ทำให้เกิดสัญญาณเสียงข้ามไปยังคู่สุนทรีย์อื่นได้แม้ว่าจะมีการหุ้มแพนโลหะไว้เพื่อป้องกันสัญญาณรบกวนแล้วก็ตาม

ปัญหานี้ของการใช้สายส่งสัญญาณ หรือ คอสทอลค์ (Crosstalk) ซึ่งสัญญาณหนึ่งจะรบกวนสัญญาณบนสายอีกสายหนึ่ง สายคู่เกลียวได้ถูกออกแบบมาเพื่อลดปัญหานี้โดยการบิดคู่สายเป็นเกลียวทำให้สัญญาณรบกวนในแต่ละสายหักล้างกัน ถ้าจำนวนเกลียวต่อหน่วยยาวยิ่งมากเท่าใด จะทำให้ป้องกันคอสทอลค์ได้ดีแต่ข้อเสียคือจะทำให้ความยาวของสายเพิ่มขึ้น

ในกรณีของอุปกรณ์โทรศัพท์มือถือที่เชื่อมต่อด้วยสายไฟจริง crosstalk จะเกิดขึ้นเมื่อได้ก็ตามที่สายเหล่านี้ข้ามกันสิ่งนี้ทำให้เกิดความไม่สงบในสัญญาณซึ่งปรากฏอยู่ในหูของผู้ฟังว่าเป็นสัญญาณเสียงหรือขึ้นส่วนของคำพูด การหุ้มนวนสายไฟและการบิดอย่างแน่นหนาสามารถลดสิ่งนี้ได้อย่างมากแม้ว่าการเดินสายจะเสื่อมสภาพในที่สุดเมื่ออายุมากขึ้นซึ่งจำเป็นต้องซ่อมแซม หากผู้บริโภคประสบ crosstalk บนอุปกรณ์ เช่น โทรศัพท์มือถือ สามารถแก้ไขปัญหาได้

บทที่ 3

ขั้นตอนและวิธีการดำเนินงาน

ขั้นตอนการดำเนินงาน

1. วางแผนและแบ่งหน้าที่รับผิดชอบของสมาชิกแต่ละคนในกลุ่ม
2. ศึกษาและรวบรวมข้อมูลที่ใช้ในการทำระบบต่าง ๆ และความต้องการของระบบ
3. เริ่มออกแบบระบบการทำงานเบื้องต้น นำมาประกอบเข้าด้วยกันเพื่อถูกพร้อม
 - 3.1. ออกแบบการทำงาน Protocol ในสื่อสารรับส่งข้อมูลและ Frame Design
 - 3.2. ออกแบบรูปแบบการรับส่งข้อมูล (Flow Control)
 - 3.3. ออกแบบรูปแบบการตรวจสอบข้อมูล (Error Control)
 - 3.4. ออกแบบการแปลงคลื่นสัญญาณ (FM)
 - 3.5. ออกแบบการทำงานของระบบควบคุม Arduino
4. ทำงานในส่วนที่แต่ละคนได้รับมอบหมาย
5. ทดสอบการทำงานของแต่ละส่วนและแก้ไขปัญหา เมื่อแต่ละส่วนทำงานถูกต้องแล้วจึงนำแต่ละส่วนมารวมกัน
6. ทดสอบระบบรวมทั้งหมดและแก้ไขปัญหาที่เกิดขึ้น
7. นำปัญหาที่พบเจอ มาจดบันทึก และระบุแนวทางแก้ไขเพื่อเป็นประโยชน์ในครั้งถัด ๆ ไป
8. จัดทำเอกสารรายงานรูปเล่ม

ระยะเวลาดำเนินงาน

วันที่ 5 พฤศจิกายน 2563 ถึง 25 พฤศจิกายน 2563

การออกแบบระบบ

การออกแบบระบบการถ่ายภาพและส่งข้อมูลผ่านคลื่น FM สามารถแบ่งเป็นหัวข้อได้ ดังนี้

1. การสื่อสารระหว่างอุปกรณ์
 2. การติดตั้ง และการทำงานของกล้อง (Camera)
 3. ประมวลผลภาพด้วย Machine Learning (ML)
 4. การควบคุมคำสั่งต่างๆ
 5. Flowchart การทำงาน
-
1. การสื่อสารระหว่างอุปกรณ์
 - 1.1. เพرمข้อมูล
 - openFlag เป็นตัวอักษร “o”
 - destination name และ source name เป็นตัวอักษรอย่างละ 1 ตัว
 - frame number เป็นตัวอักษร “0” และ “1”

- Data เป็นข้อมูลตัวอักษร 2 ตัว
- Check เป็นข้อมูลตัวอักษร 1 ตัว
- endFlag เป็นตัวอักษร “0” หรือ “1” โดยสำหรับ Data Frame จะเป็น “0” ก็ต่อเมื่อไม่มีข้อมูลให้ส่งต่อแล้ว และเป็น “1” เมื่อมีข้อมูลที่ต้องส่งในเฟรมต่อไปอีก และสำหรับ ACK Frame จะมี endFlag เป็น “0” เสมอ

OpenFlag	Dst	Src	Frame No.	Data	Check	EndFlag
1 Byte	1 Byte	1 Byte	1 Byte	2 Bytes	1 Byte	1 Byte

รูปที่ 3.1 Frame Design ของ Data Frame

OpenFlag	Dst	Src	Frame No.	Check	EndFlag
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

รูปที่ 3.2 Frame Design ของ ACK Frame

1.2. Flow Control

เลือกใช้เป็น Stop-and-Wait ARQ

- 1.2.1. ผู้ส่งข้อมูลจะต้องทำการสร้างเฟรมและสำเนาเฟรมข้อมูลที่จัดส่งไปเอาไว้ก่อนจนกว่าผู้ใช้จะยืนยันว่าได้รับเฟรมข้อมูลนั้นแล้ว
- 1.2.2. ผู้รับจะส่งเฟรม acknowledgment (ACK) มาให้กับผู้ส่งถ้าได้รับเฟรมที่ถูกต้อง โดยจะบอกว่าเฟรมถัดไปที่ต้องการรับมีเลขลำดับเท่าใด
- 1.2.3. ถ้าผู้รับข้อมูลได้รับเฟรมข้อมูลที่มีความผิดพลาด หรือไม่ต้องการ เฟรมที่รับมาันจะถูกทิ้งไป
- 1.2.4. ผู้ส่งจะมีการกำหนดเวลาเอาไว้หลังจากส่งเฟรมข้อมูลออกไปแล้ว ถ้าไม่ได้รับเฟรม ACK กลับมาในเวลาที่กำหนด ผู้ส่งจะทำการส่งเฟรมข้อมูลนั้นกลับไปอีกครั้งหนึ่ง

ผู้ส่งจะทำการส่งข้อมูลที่ลงทะเบียนและรอการตอบรับด้วยเฟรม ACK จากผู้รับโดยมี timeout ที่ 1500 มิลลิวินาที หากไม่มีการตอบรับกลับมาหรือเฟรม ACK มีความผิดพลาด ผู้ส่งจะทำการวนส่งเฟรมให้ใหม่อีกครั้งแล้วทำการรอซ้ำต่อไปจนกว่าเฟรม ACK ที่ได้รับจะถูกต้องแล้วจึงเริ่มส่งเฟรมต่อไปดังส่วนของโค้ดต่อไปนี้

```

while (true) //Send & "Wait"
{
    for (int i = 0; i < frame.length(); i++) //Send
    {
        this->tx->sendFM_noDelay(frame[i]);
    }
    long timer = millis();
    bool okAck = false;

    while (millis() - timer < TIMEOUT) //Wait for ACK
    {
        String ackFrame = this->rx->receiveStringFM(6); //receive ACK
        Serial.println("Received ACK FRAME: " + String(ackFrame));
        if (this->approveAckFrame(ackFrame)) //Prep to exit the "Wait"
        {
            Serial.println("Good ACK: PROCEED");
            frameNo == "0" ? frameNo = "1" : frameNo = "0"; //change frame number
            textData = textData.substring(2);
            okAck = true;
            break;
        }
    }

    if (okAck) //Exit "Wait" part
    {
        break;
    }
}

```

รูปที่ 3.3 ส่วนของโค้ดในการส่งข้อมูลด้วยโปรโตคอล Stop-and-Wait ARQ

ในส่วนของผู้รับ จะทำการรับเฟรมข้อมูลเข้ามาแล้วตรวจสอบความถูกต้องของเฟรมข้อมูล ถ้าเฟรมที่ได้รับถูกต้อง จะทำการรับเฟรมมาเลือกนำส่วนที่เป็นข้อมูลไปเก็บไว้และส่ง ACK กลับไปให้ผู้ส่ง เมื่อได้รับเฟรมครบแล้วดังสังเกตได้จากว่า ENDFLAG เป็น “0” จะจบการรับข้อมูล ดังส่วนของโค้ดต่อไปนี้

```

if (this->approveDataFrame(frame))
{
    Serial.println("Good Frame: " + String(frame));
    this->ackNo == "0" ? this->ackNo = "1" : this->ackNo = "0"; //Change Ack Number

    for (int i = 4; i < 6; i++) //store incoming data
    {
        if (frame[i] != '~')
            this->allReceiving += frame[i];
    }

    String ackFrame = this->makeAckFrame(ackNo, "0", destName);
    Serial.println("ACK FRAME: " + String(ackFrame));
    for (int i = 0; i < ackFrame.length(); i++)
    {
        this->tx->sendFM_noDelay(ackFrame[i]);
    }

    if (frame[7] == '0') //End Of Transmission
    {
        this->ackNo = "0";
        //STORE DATA HERE
        Serial.println("----- All Receiving: " + String(this->allReceiving) + " -----");
        this->allReceiving = "";
    }
}

```

รูปที่ 3.4 ส่วนของโค้ดในการรับข้อมูลด้วยโปรโตคอล Stop-and-Wait ARQ

1.3. Error Detection and Error Control

เลือกใช้วิธีการตรวจจับข้อผิดพลาดโดยวิธี Checksum แล้วนำค่ามา mod 2 โดยจะเริ่มนับ Checksum ตั้งแต่ open flag ไปจนถึง data 2 ไปต่อในกรณีของเฟรมข้อมูล และนับจาก open flag ไปถึง frame number ในกรณีของเฟรม ACK และมีการควบคุมแก้ไขข้อผิดพลาดโดยการส่งเฟรมข้อมูลที่ผิดพลาดไปใหม่อีกรังหนึ่งโดยมีโค้ดที่ใช้ในการตรวจสอบความถูกต้องดังนี้

```

int sum = 0;
for (int i = 0; i < toSend.length(); i++)
{
    sum += int(toSend[i]);
}
sum = sum % 2;

```

รูปที่ 3.5 ส่วนของโค้ดในการตรวจสอบความถูกต้องของข้อมูล

- 1.4. รูปแบบการส่งข้อมูล
ใช้รูปแบบการส่งข้อมูลผ่านคลื่นสัญญาณแบบ FM เพื่อสื่อสารระหว่าง PC1 และ PC2
- 1.5. การเข้ารหัสและการถอดรหัสข้อมูล
สัญญาณ Digital ใช้การเข้ารหัสและการถอดรหัสข้อมูลแบบ 2-FSK และสัญญาณ Analog ใช้การเข้ารหัสและการถอดรหัสข้อมูลแบบ FM
- 1.6. 2-FSK Modulation
การ modulation เป็นการเข้ารหัสข้อมูล แล้วส่งข้อมูลทำการเข้ารหัสผ่านคลื่นต่างความถี่
 1.6.1. การกำหนดความถี่และ encoding rule ของ 2-FSK Modulation
 1.6.1.1. Encoding rule
 $0 = 5 \text{ cycles / baud}$
 $1 = 10 \text{ cycles / baud}$
- 1.6.1.2. Partition = 1 bit / baud
 1.6.1.3. Baud rate = 100 baud
 1.6.1.4. Bit rate = 100 bps

```

for (int i = 0; i < NUM_FREQ; i++)
{
    freq[i] = ((i + 1) * 5) * FREQ_DIFF;
    freqDelay[i] = (1000000 / freq[i]) / NUM_SAMPLE ;
}
    
```

รูปที่ 3.6 ส่วนของโค้ดที่เป็นการกำหนดความถี่และเวลาการส่งคลื่น

1.6.1.5. คลื่น Digital ทำการ sampling สัญญาณออกเป็น 4 sample จากมุม 4 องศา คือ 0, 90, 180 และ 270 องศา

```

for (int i = 0; i < NUM_SAMPLE; i++)
{
    s[i] = sin(DEG_TO_RAD * 360.0 / NUM_SAMPLE * i);
    s_DAC[i] = s[i] * 2047.5 + 2047.5;
}
    
```

รูปที่ 3.7 ส่วนของโค้ดที่ทำการ sampling รูปแบบการส่งคลื่นออกเป็น 4 ส่วน

1.6.2. การแปลงข้อมูลเป็นข้อมูล Digital และการเข้ารหัสข้อมูลแบบ 2-FSK
 ก่อนการส่งข้อมูลจะนำข้อมูล Analog ที่จะทำการส่งมาทำการแปลงเป็น¹
 ข้อมูล Digital โดยการเริ่มเข้ารหัสและส่งข้อมูล จาก LSB ไป MSB ของข้อมูล
 แต่ละตัวอักษร นำข้อมูลที่ได้มาเข้ารหัสตาม Encoding rule แล้วทำการส่ง²
 คลื่นออกไป ตามลักษณะของคลื่นที่ได้จากการ sampling คลื่นที่ส่งออกไป
 นั้นจะเป็นการส่งคลื่นที่มีขนาด 8 baud / byte เท่ากับ 1 ตัวอักษร

```
void FM_TX::transmit(char in)
{
    int input[8];
    for( int i = 0; i < 8; i++)
        input[i] = (in >> i) & B0001;

    for (int k = 0; k < 8; k++)
        for (int cycle = freq[input[k]] / FREQ_DIFF; cycle > 0; cycle--)
            for (int sample = NUM_SAMPLE - 1; sample >= 0; sample--)
            {
                setVoltage(S_DAC[sample]);
                delayMicroseconds(freqDelay[input[k]]);
            }
}
```

รูปที่ 3.8 ส่วนของโค้ดที่ทำการส่งคลื่น digital ออกไป 1 ตัวอักษร

1.7. 2-FSK Demodulation

การ demodulation เป็นการรับข้อมูลสัญญาณ FM ที่ได้จากเสาสัญญาณที่ผ่าน amplifier มาทำการถอดรหัสออกเป็นข้อมูล

```
int prev = 0;
int count = 0;

uint16_t data = 0;
uint16_t bit_check = 0;

bool check_baud = false;
bool check_amp = false;

uint32_t baud_begin = micros();

String message = "";
```

รูปที่ 3.9 ส่วนของโค้ดที่เป็นตัวแปรสำคัญในการถอดรหัส

1.7.1. เมื่อมีสัญญาณ analog เข้ามาจะรับเข้าผ่านขา A2 ของ adruino UNO3 แล้วนำข้อมูลที่ได้มาทำการเช็คว่าค่าที่รับเข้ามาเป็นค่า V peek ของคลื่น และค่าที่รับมาก่อนหน้ายังไม่อยู่ในช่วง V peek ของคลื่น แสดงว่าเป็นการเจอคลื่นลูกนี้เป็นครั้งแรก และทำการนับจำนวนลูกคลื่นนั้น

```
int8_t FM_RX::isPeek(uint16_t val)
{
    if (val <= 200) //Using 600 with amplifier
        return 1;
    else
        return 0;
}
```

รูปที่ 3.10 ส่วนของโค้ดที่การเช็คว่าคลื่นที่รับเข้ามาอยู่ในช่วงไหน

```
int tmp = isPeek(analogRead(A2));

if (tmp == 1 and prev == 0 and !check_amp) // check amplitude
{
    check_amp = true; // is first max amplitude in that baud
    if (!check_baud)
    {
        baud_begin = micros();
    }
}
```

รูปที่ 3.11 ส่วนของโค้ดที่เป็นการเช็คการเจอคลื่นครั้งแรกและการเริ่มต้นของแต่ละ baud

```
if (tmp == 0 and prev == 1 and check_amp) {
    count++;
    //Serial.println(tmp);
    check_baud = true;
    check_amp = false;
}
prev = tmp;
```

รูปที่ 3.12 ส่วนของโค้ดที่ทำการนับลูกคลื่นเมื่อมีการเจอลูกคลื่นครั้งแรก

1.7.2. เมื่อค่าของคลีนที่รับมาเป็นค่าที่ไม่ได้อยู่ในช่วง V peek และมีการเช็คว่ามีการเริ่ม baud นั้นมาแล้วจนครบช่วงเวลาของแต่ละ baud จะนำข้อมูลของลูกคลีนที่นับได้ใน baud นั้น มาทำการถอดรหัส ตาม encoding rule และทำการ shift bit จาก LSB ไป MSB และเมื่อข้อมูลที่ได้ครบ 8 bits แสดงว่าได้ข้อมูลมา 1 ตัวอักษร และนำตัวอักษรไปเก็บไว้ในข้อมูลที่เป็นข้อความ

```
if (tmp == 0 and check_baud) {
    if (micros() - baud_begin > 9800 )
    {

        int dt = ((int(floor((count) / 5.0)) - 1) & 1);
        uint16_t last = dt << (bit_check);
        data |= last;

        bit_check++;
        if (bit_check == 8) // 8 bits
        {
            if (data != 0)
                message += char(data);
            if (message.length() == maxLength)
                break;
            data = 0;
            bit_check = 0;
        }
        check_baud = false;
        count = 0;
    }
}
```

รูปที่ 3.13 ส่วนของโค้ดที่เป็นการถอดรหัสข้อมูล

2. การติดตั้ง และการทำงานของกล้อง (Camera)

2.1. การติดตั้ง

2.1.1. ประกอบฮาร์ดแวร์ของกล้องและเชื่อมต่อสายไฟ

2.1.2. ทดสอบอัปโหลดโค้ด Arduino (จากไฟล์ Arduino3/Camera/Camera.ino) ที่มีการแก้ไข ลดขนาดไฟล์รูปถ่ายให้เล็กลงจาก 320x240px ลดลงเหลือ 160x120px เนื่องจากรูปที่ได้จะพอดีกับเฟรมที่ต้องการใช้ในการประมวลผล เพิ่มความเร็วและประสิทธิภาพในการถ่ายรูป



รูปที่ 3.14 ตัวอย่างภาพที่ถ่ายได้จากกล้อง (160x120px)

2.1.3. Arduino ที่ต่อ กับ Raspberry pi เชื่อมต่อผ่าน Serial port 1,000,000 Baud

```
573 //enable serial
574 UBRR0H = 0;
575 UBRR0L = 1; //0 = 2M baud rate. 1 = 1M baud. 3 = 0.5M. 7 = 250k 207 is 9600 baud rate.
576 UCSR0A |= 2; //double speed aysnc
```

รูปที่ 2.1.3 แสดงการตั้งค่า Buad ที่เชื่อมต่อของ Arduino

(จากไฟล์ Arduino3/Camera/CameraController.cpp line 575)

2.1.4. เมื่อ Arduino เริ่มทำงานจะทำการถ่ายรูปไปเรื่อย ๆ และส่งข้อมูลมาทีละ Byte ผ่านทาง Serial port โดยจะมี flag เปิดเป็นคำว่า “*RDY*” สามารถเปลี่ยนได้จากในโค้ด

```
594 while (!(PIND & 8)); //wait for high -> vsync
595 StringPgm(PSTR("*RDY*"));
596 while ((PIND & 8)); //wait for low -> vsync
```

รูปที่ 3.15 แสดงการตั้งค่า flag เปิดของรูปภาพ
(จากไฟล์ Arduino3/Camera/CameraController.cpp line 595)

- 2.1.5. ตั้งค่าขนาดรูปภาพที่ต้องการถ่าย ซึ่งในที่นี้ตั้งค่าเป็น 160x120px

```
--  
14 void loop() {  
15     capture(160, 120);  
16 }
```

รูปที่ 3.16 แสดงการตั้งค่าขนาดของรูปภาพ
(จากไฟล์ Arduino3/Camera/Camera.ino line 15)

- 2.1.6. เขียนโปรแกรมเพื่อเชื่อมต่อ Serial port ด้วยภาษา Python โดยใช้ Library “serial”

```
6 import serial
```

รูปที่ 3.17 เรียกใช้ library serial
(จากไฟล์ PC2/server.py line 7)

- 2.1.7. เปิดการเชื่อมต่อกับ Serial port และตั้งค่า Port, baudrate (Setting ต่าง ๆ จะถูกเรียกใช้งานจากไฟล์ Config (PC2/config.json))

```
serial_Arduino3 = serial.Serial(  
    port=config['port_Arduino3'],  
    baudrate=config['baudrate_Arduino3'],  
    parity=serial.PARITY_NONE,  
    stopbits=serial.STOPBITS_ONE,  
    bytesize=serial.EIGHTBITS,  
)
```

รูปที่ 3.18 ตั้งค่าการเชื่อมต่อ Serial
(จากไฟล์ PC2/server.py line 186)

- 2.1.8. อ่าน Data ที่ส่งผ่าน Serial port จนกว่าจะเจอคำว่า “*RDY*” โดยต้อง decode ข้อมูลที่ได้เนื่องจากการส่งข้อมูลผ่าน Serial นั้นเป็นจะถูกแปลง ข้อมูลเป็น byte

- 2.1.8.1. เมื่อเจอคำว่า *RDY* แล้วให้อ่านข้อมูลอีก 160x120 Byte (ข้อมูลของ รูปภาพ)
2.1.8.2. แปลงข้อมูลที่ได้เป็นรูปภาพ โดยใช้ Image จาก Library PIL
2.1.8.3. บันทึกรูปภาพที่ได้ เพื่อรอนำไป Process ต่อไป

```
def read_image(serial, x):
    #print("wait for arduino rdy")
    serial.flush()
    serial.read_until('*RDY*'.encode('utf-8'))
    #print("captured")
    serial.write(str.encode('X'))
    data = serial.read(width * height)
    _image = Image.frombytes('P', (width, height), data)
    _image = _image.transpose(Image.ROTATE_270)
    _image.save('./{}.bmp'.format(x))
    #print('save image')
```

รูปที่ 3.19 การบันทึกรูปภาพจาก Data
(จากไฟล์ PC2/Image_processing.py line 83)

3. วิธีการประมวลผลภาพด้วย Machine Learning (ML)

นำรูปภาพที่ถ่ายจากกล้อง นำไปเข้า model ที่ train ไว้ ภายใน model เป็นการใช้หลักการของ Convolutional Neural Network (CNN) โดยจะมี input เป็นรูปภาพขนาด $160 * 120$ (height * width) และ มี output ออกมา 16 ค่า คือค่าความน่าจะเป็นของรูปแต่ละแบบ

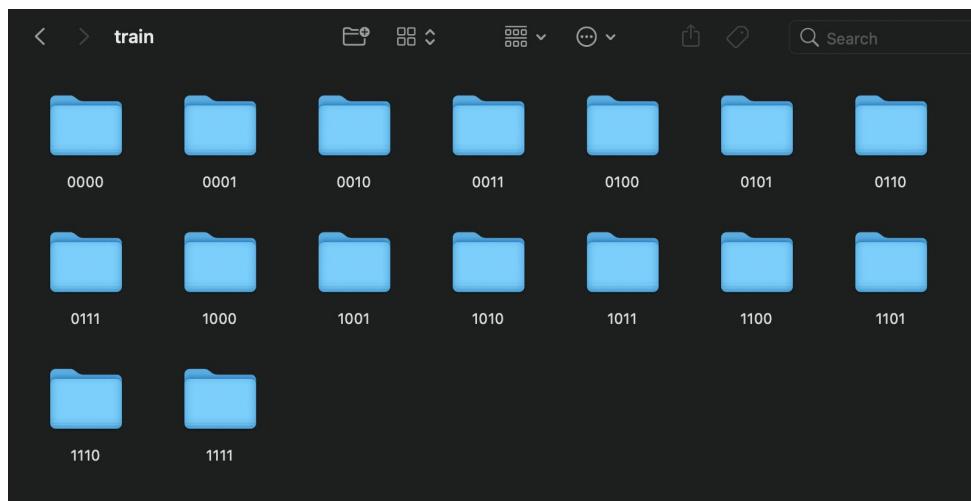
ขั้นตอนการเตรียม data set มีดังนี้

3.1 แบ่ง data ออกเป็น 3 ส่วน โดยแบ่งเป็น train 60%, validation 20%, test 20%



รูปที่ 3.20 การแบ่งโฟลเดอร์ชุดข้อมูล

3.2 แต่ละ folder จะแบ่งออกเป็น 16 folder ตาม class ที่ต้องการจำแนก



รูปที่ 3.21 การแบ่งโฟลเดอร์ย่อยตาม class ที่ต้องการจำแนก

รายละเอียดวิธีการประมวลผลภาพด้วย Machine Learning (ML) มีดังนี้

3.1. Import library ที่จะใช้ (ดังรูปที่ 3.5)

Source for train model for images recognition ☀

- This is the part of Data Communications (01076007) in KMITL

```
In [1]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
from matplotlib import pyplot as plt
import tensorflow as tf
import joblib as jl
import pandas as pd
import numpy as np
import cv2
import os

print(f'numpy version : {np.__version__}')
print(f'pandas version : {pd.__version__}')
print(f'tensorflow version : {tf.__version__}')

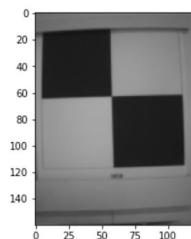
numpy version : 1.18.5
pandas version : 1.1.4
tensorflow version : 2.3.1
```

รูปที่ 3.22 การ import library

3.2. ทดลองนำเข้า, แสดงผลรูปภาพ และ import data set ต่าง ๆ (ดังรูปที่ 3.6)

```
In [2]: path_test_images = '../ML/data_set_final/train/0110/0110_1_672372.bmp'
img = image.load_img(path_test_images)
plt.imshow(img)
```

```
Out[2]: <matplotlib.image.AxesImage at 0x7feb385fa0d0>
```



```
In [3]: path_test_images = cv2.imread(path_test_images)
path_test_images.shape
```

```
Out[3]: (160, 120, 3)
```

```
In [4]: train = ImageDataGenerator(rescale=1/255)
validation = ImageDataGenerator(rescale=1/255)
train_dataset = train.flow_from_directory('../ML/data_set_final/train/',
                                         target_size=(160, 120),
                                         batch_size=3)
validation_dataset = validation.flow_from_directory('../ML/data_set_final/validation/',
                                         target_size=(160, 120),
                                         batch_size=3)
```

```
Found 864 images belonging to 16 classes.
Found 197 images belonging to 16 classes.
```

รูปที่ 3.23 ทดลองนำเข้า, แสดงผลรูปภาพ และ import ชุดข้อมูลต่าง ๆ

3.3. แสดงผล class ต่าง ๆ ตาม data set ที่เอาไว้สำหรับ train model (ดังรูปที่ 3.7)

```
In [5]: train_dataset.class_indices
```

```
Out[5]: {'0000': 0,
'0001': 1,
'0010': 2,
'0011': 3,
'0100': 4,
'0101': 5,
'0110': 6,
'0111': 7,
'1000': 8,
'1001': 9,
'1010': 10,
'1011': 11,
'1100': 12,
'1101': 13,
'1110': 14,
'1111': 15}
```

รูปที่ 3.24 แสดงผล class ต่าง ๆ ตามชุดข้อมูล

3.4. สร้าง model ตามที่ได้ออกแบบไว้ (ดังรูปที่ 3.8)

```
In [6]: model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape = (160, 120, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation='relu',input_shape = (160, 120, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.15),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.Dense(16,activation='softmax')
])
```

รูปที่ 3.25 การสร้าง model

3.5. Compile และ Train model (ดังรูปที่ 3.9)

```
In [8]: model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy'])
```

```
In [9]: model.fit(train_dataset,
    steps_per_epoch=3,
    epochs=25,
    validation_data=validation_dataset)
```

รูปที่ 3.26 การ compile และ train model

3.6. Save model (ดังรูปที่ 3.10)

```
In [11]: model.save('model_datacom_test')
```

รูปที่ 3.27 การบันทึก model ที่ได้รับ

4. การควบคุมคำสั่งต่างๆ

4.1. การทำงานของระบบ การออกแบบระบบต้องการให้มีความเสถียรและความเร็วสูงสุด จึงทำเอาระบบที่มีความซับซ้อน เช่นการใช้ Threading เข้ามาเป็นส่วนการทำงานหลัก เพื่อให้โปรแกรมสามารถทำงานทุกส่วนพร้อมกันได้ และใช้ภาษา Python เป็นหลัก เนื่องจากมีความยืดหยุ่นในการเขียนและมี library ที่ช่วยให้การทำงานได้ง่ายขึ้น

4.1.1. Raspberry PI 1 (PC1) เชื่อมต่อ กับ arduino 1 มี Thread การทำงานหลัก ๆ 3 thread โดยการเขียนโปรแกรมจะเน้นไปทางการใช้ตัวแปร global เป็นหลัก เพื่อให้ทุก function ในแต่ละ thread สามารถใช้ข้อมูลร่วมกันได้

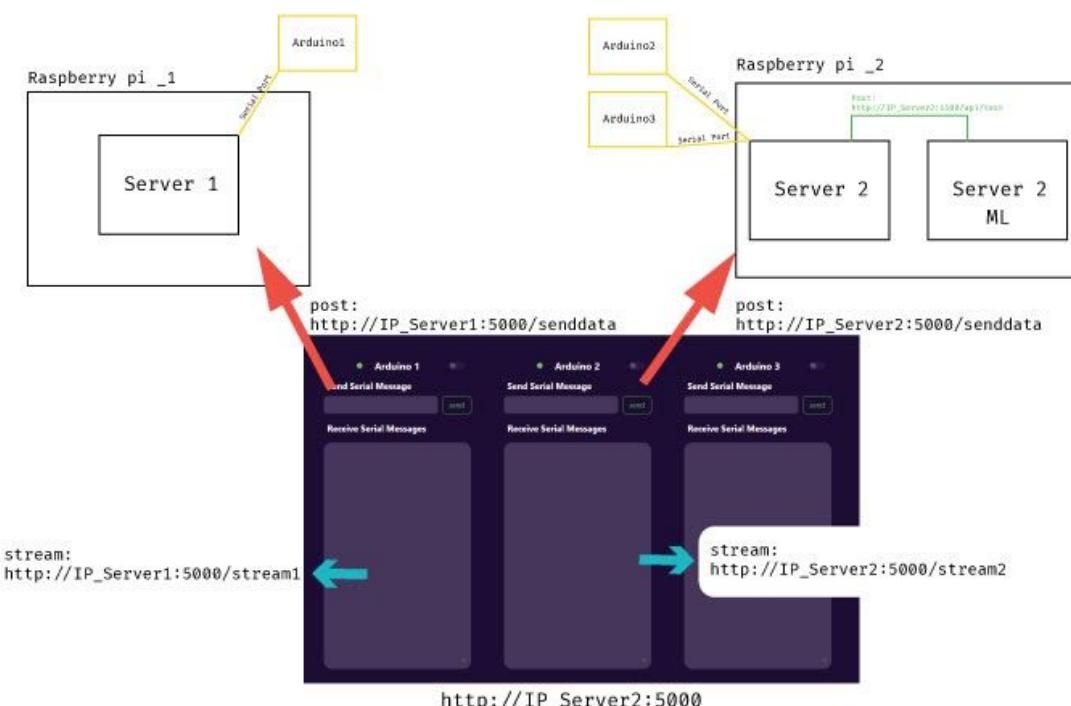
4.1.1.1. Thread ที่ 1 ทำหน้าที่คุ้ม Web server ได้แก่

4.1.1.1.1. Post /send data ทำหน้าที่ส่งข้อความไปยัง Serial port ของ arduino 1

4.1.1.1.2. Stream /stream1 ทำหน้าที่ stream data ที่ส่งมายัง PC ไปยังหน้าเว็บ

4.1.1.2. Thread ที่ 2 ทำหน้าที่รับ input จาก keyboard แล้วส่งข้อความที่ได้ยังไป arduino ผ่านทาง serial port

4.1.1.3. Thread ที่ 3 ควบคุมการเชื่อมต่อของ arduino โดยทำหน้าที่ควบคุมทั้ง read และ write



รูปที่ 3.28 การเชื่อมต่อของระบบ

- 4.2. Raspberry PI 2 เชื่อมต่อกับ arduino 2 และ arduino 3 โดยใช้หลักการเดียวกันกับ Raspberry PI 1 ที่นำเอา thread มาใช้ในการทำงาน
- 4.2.1. Server ที่ทำหน้าที่ประมวลผลภาพ (ML) จะเปิดเป็น server แยก ที่ port 5500 (http://ip_server:5500/api/test) เหตุผลที่ต้องแยกออกมาเป็นอีก server เพราะว่าจะสามารถสร้างความยืดหยุ่นให้ระบบได้มากกว่า หาก raspberry pi ตัวที่ 2 ไม่สามารถประมวลผลทั้งหมดได้ จะสามารถย้ายเอา server ไปรันไว้ที่อื่นแล้วยิง request ไปที่ server นั้นแทนได้
- 4.2.2. Server หลัก มี 4 thread
- 4.2.2.1. Thread 1 ทำหน้าที่ควบคุม web server
- 4.2.2.1.1. โดยเมื่อเข้ามายัง port 5000 ใน home directory (http://ip_server:5000/) server จะตอบกลับเป็น static ไฟล์ของหน้าเว็บ (main.html)
- 4.2.2.1.2. /senddata ทำหน้าที่ส่งข้อความไปยัง serial port ของ arduino 2
- 4.2.2.1.3. /stream2 ทำหน้าที่ stream ข้อมูลที่ได้จาก arduino 2 ไปยังหน้าเว็บไซค์
- 4.2.2.2. Thread 2 ทำหน้าที่ควบคุมการเชื่อมต่อของ raspberry pi และ arduino 2 ควบคุมทั้ง read และ write
- 4.2.2.3. Thread 3 ควบคุมการ display Led แสดงข้อมูล binary ที่กล้องสามารถ detect ได้ 12bit แบ่งเป็น left 4 bit, middle 4 bit, right 4 bit
- 4.2.2.4. Thread 4 ทำหน้าที่ควบคุมการรับคำสั่งให้กล้องหมุนและถ่ายรูป โดยมีสองแบบ คือถ่ายทั้ง 3 มุม และถ่ายเฉพาะมุม
- 4.2.2.4.1. หมุนถ่ายทั้ง 3 มุม สิ่งที่ได้จาก function จะเป็น [left, middle, right] รหัสเลขฐาน 10 ของภาพทั้ง 3 มุม โดยกล้องจะเริ่มจากการหมุนไปฝั่งซ้าย และทำการถ่ายรูป โดยจะมีการ flush รูปทั้ง 3 รูป (จากการทดสอบแล้ว 3 รูปแรกมักเป็นรูปที่ถ่ายแล้วใช้ไม่ได้) และทำการถ่ายรูปที่ 4 เพื่อนำมาประมวลผล delay 0.5 วินาที และทำการซ้ำอีกครั้ง นำคำตอบทั้งสองครั้งมาตรวจสอบ หากเป็นคำตอบเดียวกัน ให้หมุนไปยังอีกสองมุม แต่หากไม่เหมือนกัน จะทำการถ่ายซ้ำ จนกว่าจะได้คำตอบตรงกัน
- 4.2.2.4.2. หมุนไปถ่ายยังมุมที่ระบบ สิ่งที่ได้จาก function จะเป็นค่าสี่ 16 จุด และอีก 4 ค่า เป็นค่าเฉลี่ย

4.3. คำสั่งต่าง ๆ

4.3.1. capture หาก server 2 ได้รับคำสั่งนี้ จะทำการถ่ายภาพ ทั้ง 3 มุม โดยการส่งงานนั้นสามารถทำได้ 2 วิธี คือสั่งจาก arduino 1 หรือสั่งจากหน้าเว็บ โดยเมื่อระบบทำงานเสร็จตามคำสั่งจะทำการส่ง XXXX โดยตัวแรกเป็น flag เพื่อให้ง่ายต่อการ debug XXX จะเป็นเลขฐาน 16 ของค่าทั้ง 3 มุม ทำให้ส่งข้อมูลเพียง 4 byte กลับมายัง arduino 1 เมื่อ server ฝั่ง pc 1 ได้รับข้อความจะทำการตรวจสอบ flag และความยาวของข้อความที่ได้รับ หากเข้าเงื่อนไข จะทำการ decode ข้อมูลกลับเป็น binary ของรูปภาพต่อไป

```
*** Answer from PC2 ***
-45 Degree: 0001
0 Degree: 0010
45 Degree: 0011
*** ----- ***
```

รูปที่ 3.29 แสดงการ decode ข้อความ “X123” ในฝั่ง PC1

- 4.3.1.1. การสั่งจาก arduino หากพิมพ์คำว่า capture arduino 1 จะทำการส่งข้อความนั้นมา yัง arduino 2 จะทำการส่งข้อความที่ได้ไปยัง serial , server จะสั่งให้กล้องหมุนและบันทึกรูปภาพตามระบบต่อไป
- 4.3.1.2. หากสั่งจากหน้าเว็บไซต์ http://ip_server2:5000/ (ใช้สำหรับการ debug) พิมพ์คำว่า “TEST” ไปยัง Serial 2 บนหน้าเว็บระบบจะสั่งให้มีการถ่ายรูปเช่นกัน
- 4.3.2. พิมพ์เลขฐาน 2 ของ 0-15 ตัวอย่าง เช่น 0100 ระบบจะทำการเปลี่ยนข้อมูลที่ได้รับจากฐาน 2 เป็นฐาน 16 เพื่อลดขนาดของข้อมูลที่ส่งให้เหลือ 1 byte ไปยัง serial ของ arduino 1 จะเป็นการสั่งให้กล้องหมุนไปยังมุมที่รูปนั้นอยู่ หากเลขฐานสองที่พิมพ์มาไม่มีอยู่ในหน่วยความจำที่ระบบบันทึกไว้ หรือยังไม่มีการสั่งให้ถ่ายรูปทั้งหมด (capture) ระบบจะไม่ทำการคำสั่งนี้ แต่ถ้ามีการสั่งให้ถ่ายรูปแล้ว แต่มีการเปลี่ยนรูปที่ในมุมนั้น ระบบจะสั่งให้กล้องหมุนไปยังตำแหน่งเดิมที่รูปนั้นอยู่ แล้วทำการถ่ายรูปใหม่ทุกครั้งส่งกลับเป็นค่า 20 ค่า เมื่อระบบทำงานเสร็จ จะทำการส่ง *ค่าสี 20 ค่าที่แปลงเป็น char* เพื่อลดขนาดนั้นข้อมูลลง เนื่องจากรูปที่ถ่ายได้เป็น gray color มีค่าสีตั้งแต่ 0-255 ทำให้สามารถเก็บข้อมูลได้ภายใน 1 byte และใช้ flag เปิดและปิดเป็น * เพื่อให้ง่ายต่อการตรวจสอบและ decode ในฝั่ง PC1 ต่อไป เมื่อ PC1 ได้รับข้อความจะทำการตรวจสอบความยาวและ flag หากเข้าเงื่อนไข จะทำการ decode ข้อมูลที่ได้ โดยจะแสดงพิกัดที่นำเอาค่าของจุดสีมาแสดงด้วย (fixed พิกัดไว้)

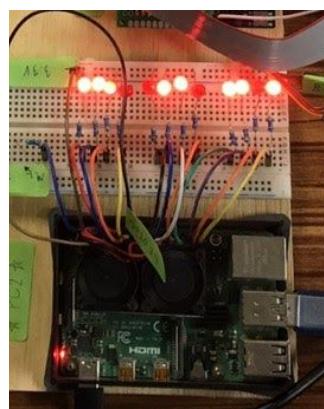
```

*** Answer from PC2 ***
(40,20): 49
(44,24): 50
(48,28): 51
(52,32): 52
mean of QUADRANT 0: 53
(40,70): 54
(44,74): 55
(48,78): 56
(52,82): 57
mean of QUADRANT 1: 48
(100,20): 97
(104,24): 98
(108,28): 99
(112,32): 100
mean of QUADRANT 2: 101
(100,70): 102
(104,74): 49
(108,78): 50
(112,82): 51
mean of QUADRANT 3: 52
*** ----- ***

```

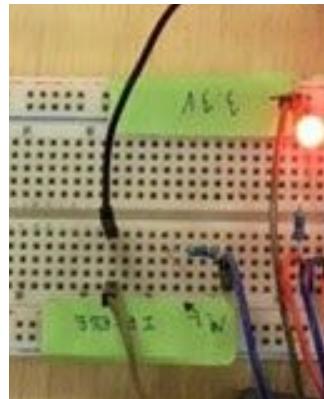
รูปที่ 3.30 ข้อความที่ถูก decode แล้ว

- 4.4. มีการแสดงผลค่าที่ได้จากทั้ง 3 มุมหรือมุมใดมุมหนึ่งอุกมาทางหลอด LED เนื่องจากใช้ Raspberry pi แทน จึงทำให้มี GPIO ให้ใช้ในการเชื่อมต่อและควบคุมอุปกรณ์ภายนอกได้ โดยการทำงานจะทำงานอยู่บน thread แยกอิสระ โดยรับจาก Array [left, middle, right] หากค่าเป็นค่าที่ไม่อยู่ในช่วง 0-15 จะแสดงเป็นสถานะ loadding เช่น [15, -1, -1] หลอดที่ 1-4 (นับจากซ้าย) จะติดทั้ง 4 ดวง หลอดที่ 5-12 จะติดเป็นสถานะ loadding (ไฟวิ่งไปเรื่อย ๆ)



รูปที่ 3.31 การแสดงผลของหลอด LED

- 4.5. ระบบเลือกการประมวลผลภาพ มีสองแบบได้แก่ใช้ if else หรือ ML โดยการเปลี่ยน mode นั้น ทำได้โดยการ สลับสายที่เชื่อมต่ออยู่กับ Raspberry pi



รูปที่ 3.32 การสลับโหมดของการประมวลผลภาพ

- 4.5.1. ในระบบ if else นั้นจะใช้วิธีการ ดึงค่า 4 จุดจากแต่ละ quadrant มาหา mean หากค่านั้นมากกว่า 80 แสดงว่าเป็นสีขาว แต่หากน้อยกว่า 80 แสดงว่า เป็นสีดำ (80 มาจากค่าแสดงที่วัดได้ ณ สถานที่ทดลอง) หากเลือกวิธีนี้ในการ ประมวลผลจะได้ทั้ง 20 ค่าสี และคำตอบว่า รูปนั้นคือรูปแบบได้
4.5.2. ใช้ ML ประมวลผล Server 2 จะทำการส่งรูปภาพไปยัง Server ML เพื่อนำไป ประมวลผลกับ model ที่ได้เตรียมไว้แล้วตอบกลับเป็นรูปแบบของรูปนั้น หากเลือกใช้วิธีนี้ ระบบจะเรียกใช้การตรวจสอบแบบ if else เองอีกรอบหนึ่ง เพื่อให้ได้ค่าสี 20 ค่าสีด้วย
5. การติดตั้งระบบต่าง ๆ ในการทดสอบทางผู้จัดทำ ได้ทดสอบระบบ Server ต่าง ๆ ใน OS Linux (arm)
- 5.1. หากไม่ต้องการแก้ไข config ต่าง ๆ ให้กำหนดค่าต่าง ๆ ดังนี้
- 5.1.1. IP Raspberry PI1: 192.168.88.17
5.1.2. IP Raspberry PI2: 192.168.88.19
- 5.2. ระบบถูกพัฒนาขึ้นมาด้วยภาษา python version 3

```
PRETTY_NAME="Debian GNU/Linux 10 (buster)"  
NAME="Debian GNU/Linux"  
VERSION_ID="10"
```

5.3. ฝั่ง PC1 (Raspberry PI 1)

5.3.1. ติดตั้ง library จาก PC1/requirements.txt ด้วยคำสั่ง

```
pip3 install -r requirement.txt
```

5.3.2. แก้ไขไฟล์ PC1/config.json เป็น port ที่เชื่อมต่ออยู่กับ arduino 1

```
{
  "port_Arduino1": "/dev/ttyACM0",
  "buadrate_Arduino1": 115200
}
```

5.3.3. รันไฟล์ PC1/terminal_pc1.py ด้วยคำสั่ง

```
python3 terminal_pc1.py
```

5.4. ฝั่ง PC 2 (Raspberry PI 2)

5.4.1. ติดตั้ง library จาก PC2/requirements.txt ด้วยคำสั่ง

```
pip3 install -r requirement.txt
```

5.4.2. แก้ไขไฟล์ PC2/config.json เป็น port ที่เชื่อมต่ออยู่กับ arduino 2, arduino 3 และ IP ของ server ประมวลผลภาพ

```
{
  "port_Arduino2": "/dev/ttyACM1",
  "buadrate_Arduino2": 115200,
  "port_Arduino3": "/dev/ttyACM0",
  "buadrate_Arduino3": 1000000,
  "server_processing": "localhost"
}
```

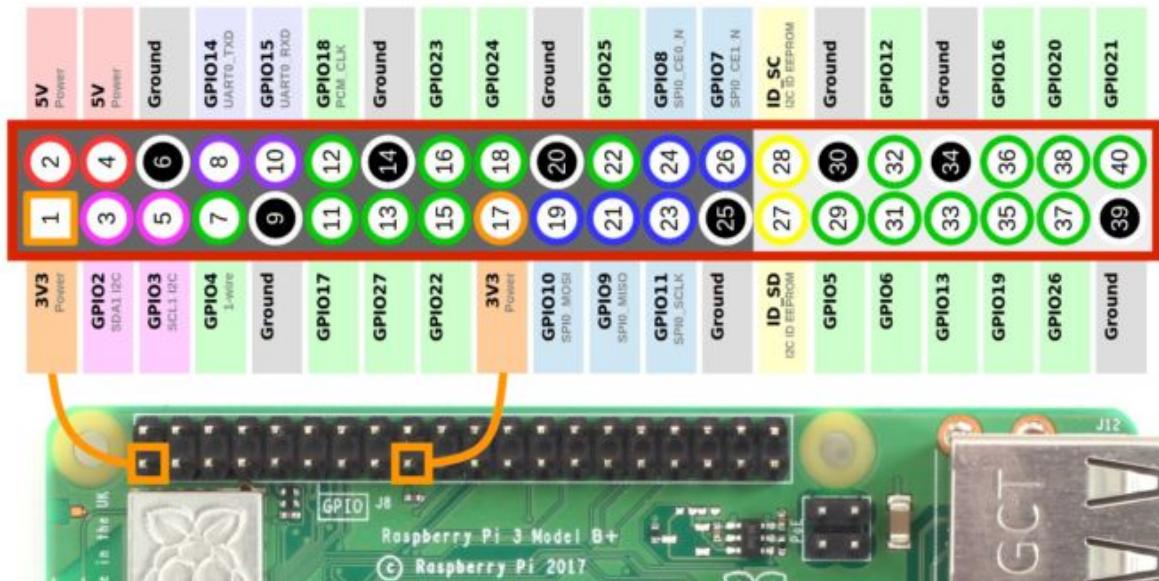
5.4.3. แก้ไขการตั้งค่าของหน้าเว็บไซต์ จากไฟล์
PC2/static/webDisplaySerial/setting.js เป็น IP ของ server ทั้ง 2 ตัว

```
const settings = { target:  
  ['http://192.168.88.17:5000',  
   'http://192.168.88.19:5000',  
   'http://192.168.88.19:5000' ] };
```

5.4.4. แก้ไขไฟล์ PC2/templates/main.html บรรทัดที่ 24 เป็น ip ของ server 1

```
var source1 = new EventSource("http://192.168.88.17:5000/stream1");
```

5.4.5. เชื่อมต่อหลอด LED จากขา GPIO 26, 19, 13, 6, 5, 11, 9, 10, 22, 27, 17,
18 โดยไล่จาก MSB ไปยัง LSB และเชื่อมต่อขา GPIO 23 ไปยัง GND หรือ
VCC เพื่อสลับ mode if else, ML



รูปที่ 3.33 pinout ของ raspberry pi 4

5.4.6. สั่งรันไฟล์ PC2/server.py เพื่อสั่งให้ server หลักทำงาน

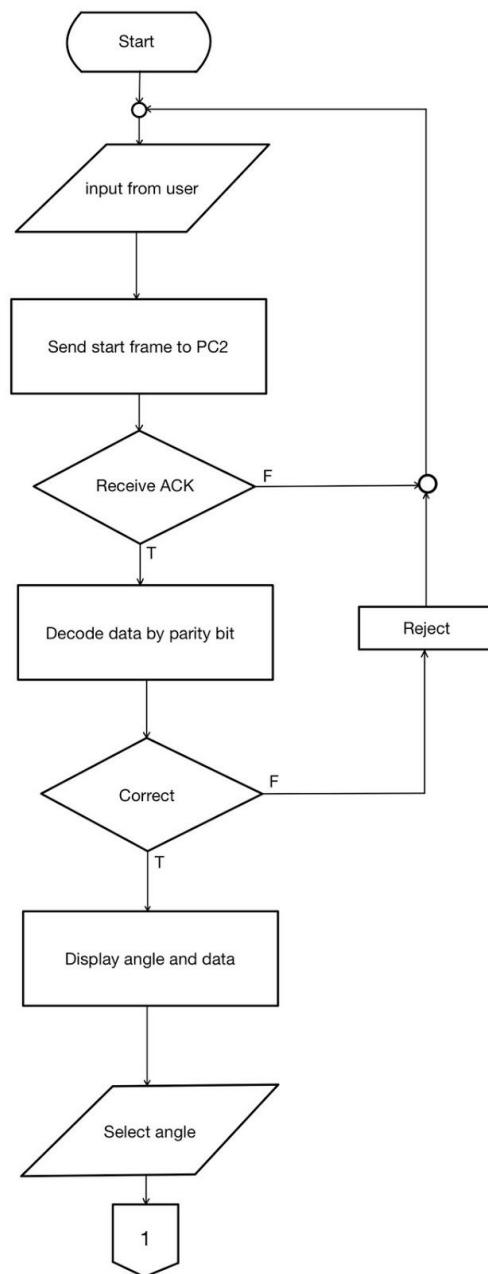
```
python3 server.py
```

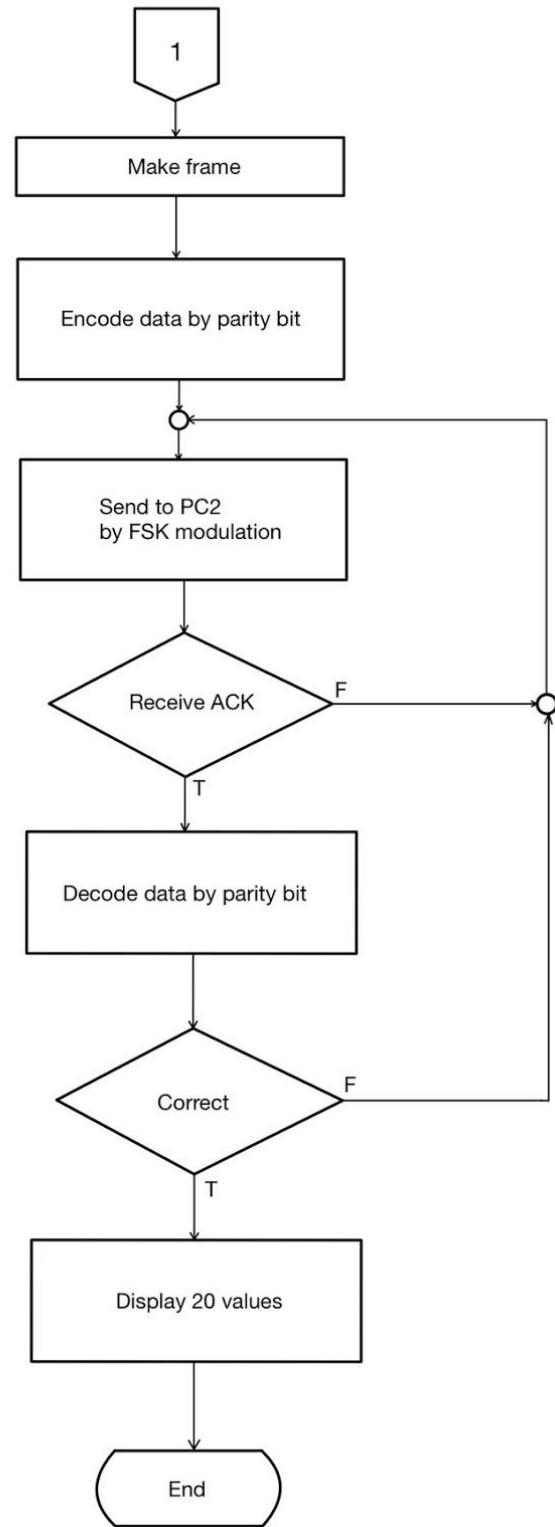
5.4.7. สั่งรันไฟล์ PC2/server_image.py เพื่อสั่งให้ server สำหรับประมวลผล ML ทำงาน

```
python3 server_image.py
```

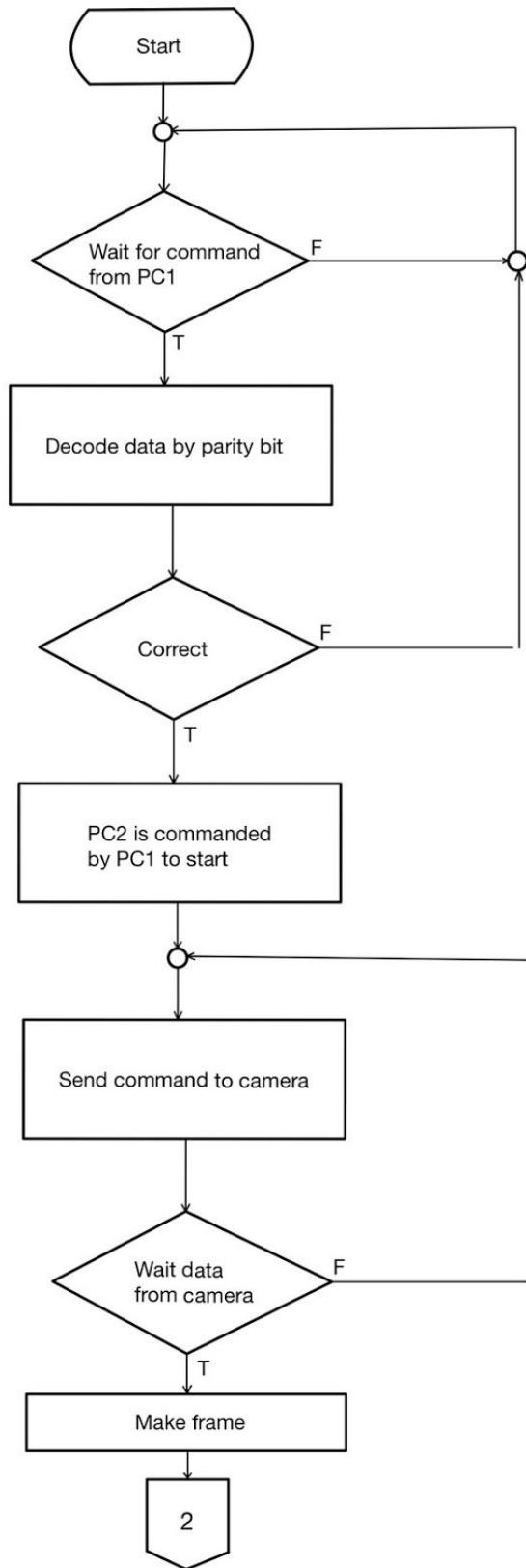
6. Flowchart การทำงาน

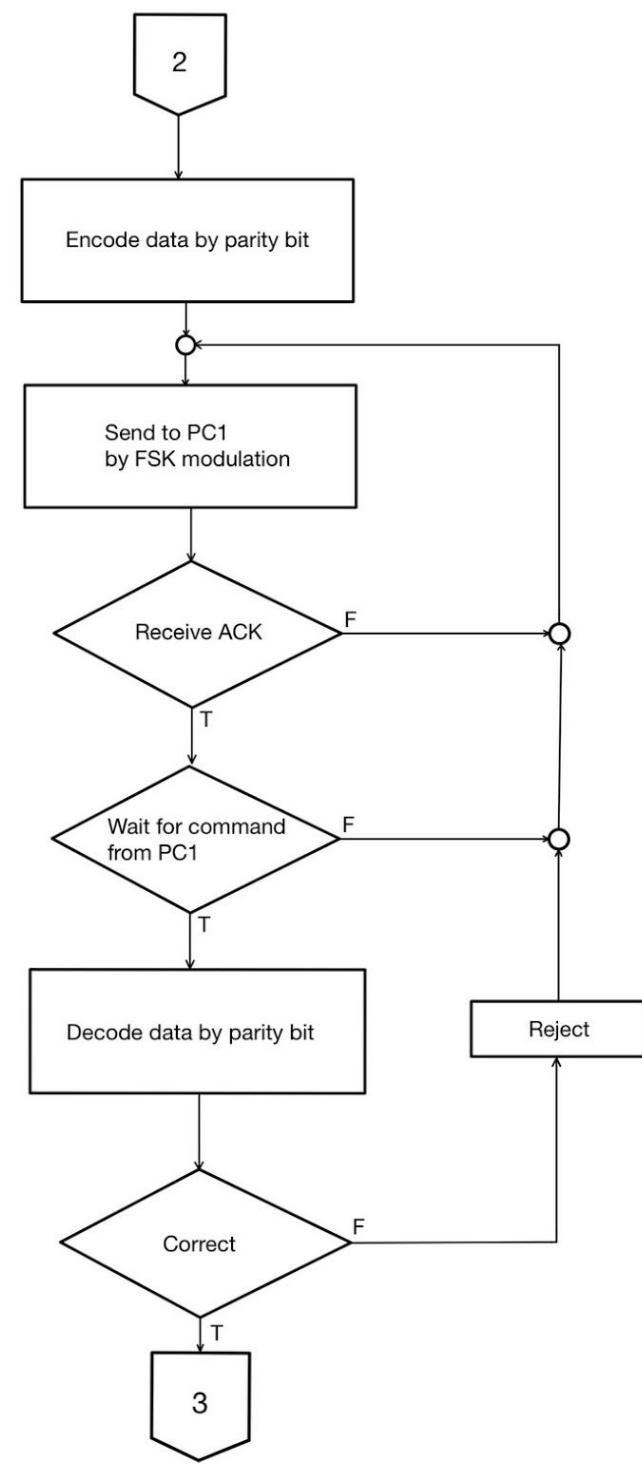
6.1. การทำงานทางผู้ใช้งานของ PC 1

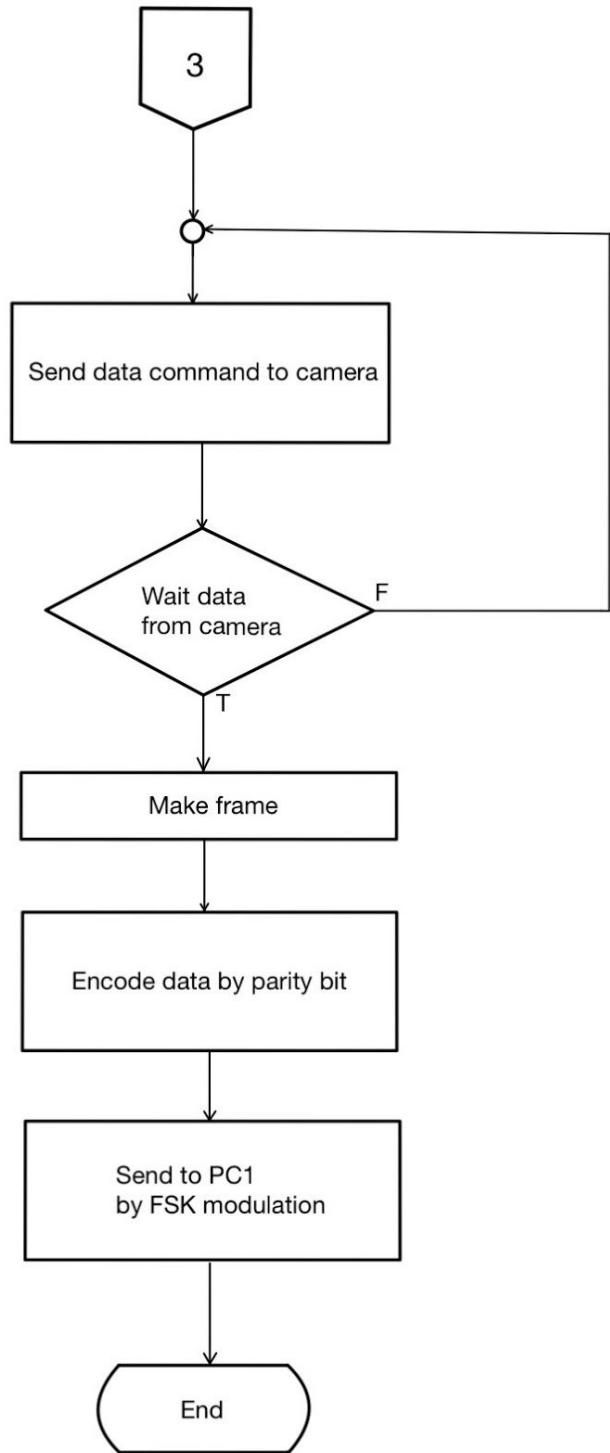




6.2. การทำงานทางผังของ PC 2







วัสดุอุปกรณ์และโปรแกรมที่ใช้ (Hardware & Software)

- ด้านฮาร์ดแวร์

○ Arduino Uno	3 ตัว
○ Raspberry Pi 4	2 ตัว
○ OV7670 Camera Module	1 ตัว
○ TEA5767 FM Stereo Radio Module	2 ตัว
○ FM Transmitter Module	2 ตัว
○ SG90 Servo Motor	2 ตัว
○ สายไฟ Jumper Wires Male to Male	40 เส้น
○ แผ่น PCB 4 x 6 cm	1 แผ่น
○ ตะเก็บบัดกรี	1 ปอนด์
○ หัวแร้งบัดกรี	1 ชิ้น
○ สายแพ 20 pin	1 เส้น
○ หัวสายแพ 20 pin	2 หัว
○ พิวเจอร์บอร์ด 65 x 85 cm	1 แผ่น
○ แผ่นไม้พลาสติก 65 x 85 x 0.5 cm	1 แผ่น
○ Hantek Digital Oscilloscope	2 เครื่อง
○ สาย Probe สัญญาณ	4 เส้น

- ด้านซอฟต์แวร์

- โปรแกรม Arduino IDE
- โปรแกรม Visual Studio Code
- โปรแกรม Hantek 6022BL
- Git

บทที่ 4

ผลการดำเนินงาน

ผลการดำเนินงานตามขอบเขตของโครงการ

จากการทดลอง มีส่วนที่สามารถทำได้ตามขอบเขตของโครงการได้แก่

1. ส่วน Raspberry Pi ตัวที่ 1 (แทน PC1)

- สามารถสั่งให้ฝั่ง RasPi 2 เริ่มต้นการทำงาน (เก็บข้อมูลจากกล้องทุกมุมภาพ)
- แสดงผลสรุปข้อมูลรหัส Binary (0000, 0001, ..., 1111) และ มุม (-45, 0, +45) ที่ได้รับจากกล้องฝั่ง RasPi 2 มาแสดงผล
- หลังจากนั้น สามารถเลือก
 - สั่งให้ RasPi 2 เก็บค่าข้อมูลภาพตามรหัส Binary ที่ได้
 - RasPi 2 หมุนกล้องไปยังภาพตามรหัส Binary ที่สั่ง
 - RasPi 2 ส่งข้อมูลจุดภาพ (ระดับสีภาพ) ตามที่กล้องหมุนไป จำนวน 20 ค่า มาแสดงผลที่ RasPi 1
 - จำนวนข้อมูล 20 ค่า มาจาก Quadrant ละ 5 ค่า ประกอบด้วย
 - ค่าระดับสีภาพจำนวน 4 จุด
 - ค่าพิกัดแต่ละจุดค่าเฉลี่ยในแต่ละ Quadrant
 - สั่งให้เริ่มต้นทำงานใหม่

2. ส่วน Raspberry Pi ตัวที่ 2 (แทน PC2)

- รอรับคำสั่งเริ่มการทำงานจาก RasPi 1 (เก็บข้อมูลจากกล้องทุกมุม พร้อมวิเคราะห์ชนิดข้อมูลรหัส Binary ภาพ)
- ขั้นตอนการทำงานของ RasPi 2 วิเคราะห์ข้อมูลชนิดข้อมูลรหัส Binary ภาพ จากกล้องในมุมต่าง ๆ แล้วส่งข้อมูลสรุปให้ RasPi 1
- รอรับคำสั่งจาก RasPi 1
 - หาก RasPi 1 สั่งให้เก็บค่าข้อมูลภาพตามรหัส Binary
 - หมุนกล้องไปมุมภาพรหัส Binary ที่กำหนดจาก RasPi 1
 - เก็บข้อมูลภาพ
 - ส่งข้อมูลจำนวนจุดภาพตามที่กล้องหมุนไปกลับมา RasPi 1
 - หาก RasPi 1 สั่งให้เริ่มต้นทำงานใหม่ จึงเริ่มทำงานใหม่ตั้งแต่ต้น

3. ส่วน Tx, Rx, Camera

- ใช้เป็น Arduino

4. ส่วนการเชื่อมต่อระหว่าง Arduino <-> Raspberry Pi

- เป็นสาย USB 2.0 Type B
-

5. Communication (ระหว่าง Raspberry Pi ตัวที่ 1 - Raspberry Pi ตัวที่ 2)

- ต้องใช้ Digital Modulation ที่เหมาะสม โดยเลือกใช้ 2-FSK
- มี Error Detection ด้วยวิธี checksum
- มีการกำหนด Frame Design ที่เหมาะสม
- มีการกำหนด Flow & Error Control โดยใช้วิธี Stop-and-Wait ARQ

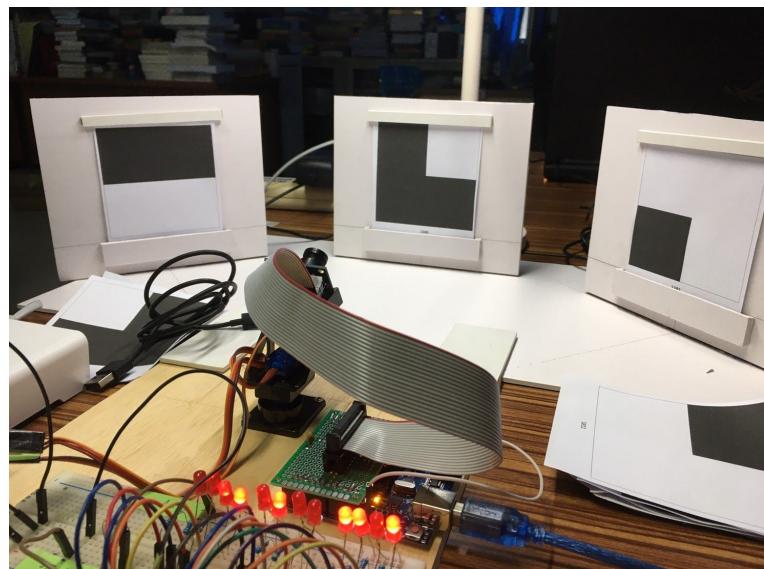
ส่วนที่ไม่สามารถทำตามขอบเขตที่ตั้งไว้ได้แก่ การส่งสัญญาณผ่าน Wireless ผ่านวิธี FM เนื่องด้วยสัญญาณรบกวนที่สูง ทั้งนี้ทางผู้จัดทำได้ทดลองหาวิธีทางแก้ปัญหาที่หลากหลายแล้วได้แก่ การทำ Band Pass Filter, การแก้ปัญหา r-slope ในซอฟต์แวร์, การคำนวณความสูงเสาระยะห่างสัญญาณ, การจัดตำแหน่งตัวส่งและตัวรับใหม่, เปลี่ยนตัวส่ง ตัวรับ และ Amplifier, เปลี่ยน DAC, ส่งสัญญาณในตอนกลางคืน กรณั้นแล้วทางผู้จัดทำก็ไม่สามารถแก้ไขปัญหาความผิดพลาดในการส่งสัญญาณผ่าน FM ในช่วงเวลาส่งงานได้

ทางผู้จัดทำจึงเปลี่ยนไปใช้การส่งโดยใช้สายต่อแทน โดยต่อสายผ่านจากสัญญาณออกของ DAC ไปเข้าที่ขาพินรับสัญญาณแทน ข้อสังเกตที่น่าสนใจคือเมื่อทำการส่งสัญญาณในตอนกลางคืน สัญญาณจะมีคลื่นรบกวนที่ต่ำกว่าตอนกลางวันเป็นอย่างมาก

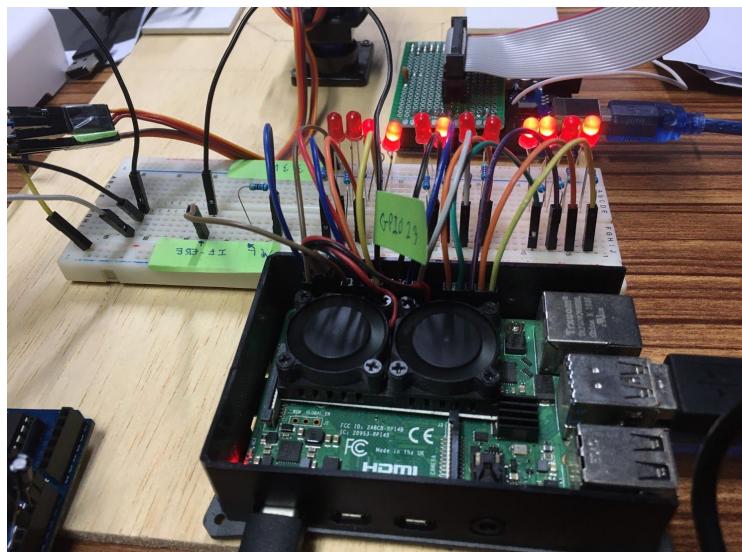
ขั้นตอนการใช้ระบบ

1. เริ่มต้นการทำงานระบบโดยการพิมพ์คำว่า capture ไปยัง Terminal (ที่มีการนำเอา Serial monitor ของ arduino มารวมไว้) ของ PC1
2. เมื่อได้รับการตอบกลับจาก Arduino 2 แล้ว จะสามารถพิมพ์ตอบกลับ เป็นเลขฐาน 2 เพื่อสั่งให้กล้องหมุนไปถ่ายยังมุมที่ภาพนั้นอยู่ [-45, 0, +45 องศา] เพื่อขอข้อมูล 20 ค่าพิกัดจุดสี่บนรูปภาพ
3. Terminal จะแสดงค่าที่ได้ที่ถูก decode แล้วเป็น 20 ค่าพร้อมกับพิกัดที่ถ่ายมา (fixed) สามารถสั่งให้กล้องหมุนไปยังองศาอื่น ๆ ต่อได้
4. สามารถสั่งงานได้จากหน้าเว็บไซต์ http://ip_server2:5000 ด้วยเช่นกัน

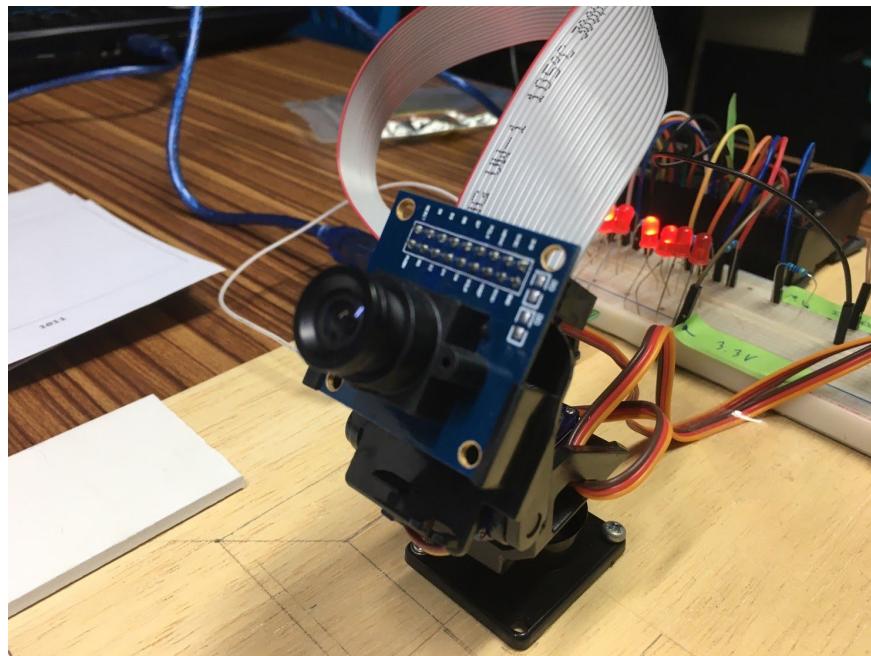
ผลการดำเนินงาน



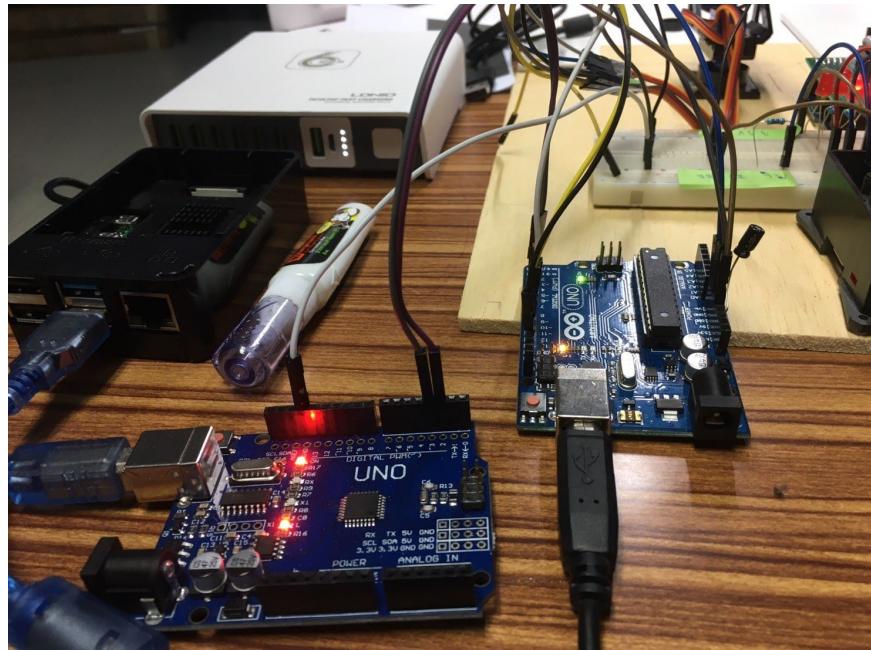
รูปที่ 4.1 แสดงการถ่ายภาพด้วยกล้อง OV7670 กับตัวอย่างรูปแบบภาพ



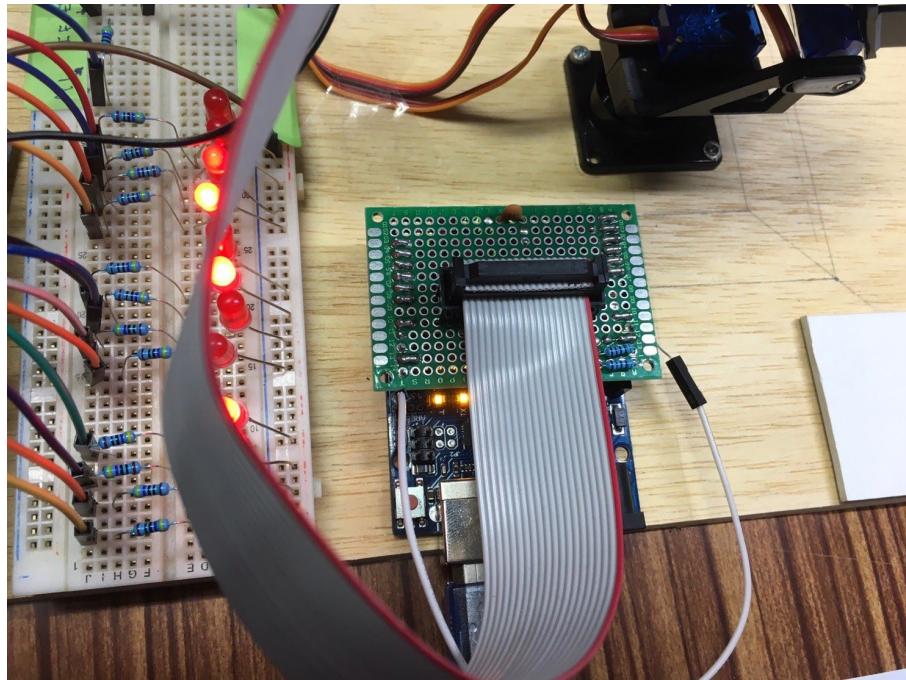
รูปที่ 4.2 แสดงไฟ LED จากข้อมูลที่กล้องอ่านได้จากแต่ละภาพ



รูปที่ 4.3 กล้อง OV7670 Module ที่ใช้ในการถ่ายภาพ



รูปที่ 4.4 แสดงการทำงานระหว่าง Arduino FM & Servo และ Arduino ของกล้อง
ผ่าน Raspberry Pi ตัวที่ 2



รูปที่ 4.5 แสดง Shield บัดกรีต่อเข้ากับ Arduino
เพื่อหลีกเลี่ยงการใช้สาย Jumper โดยไม่จำเป็น

ภาพถ่ายที่ได้จากการกล้อง OV7670 Camera Module

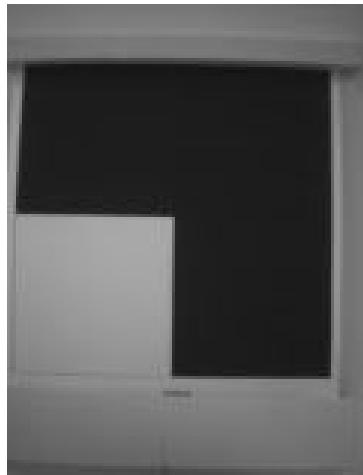
- ภาพถ่ายความละเอียด 160×120 pixels



รูปที่ 4.6 รหัสเลขฐานสอง 0000



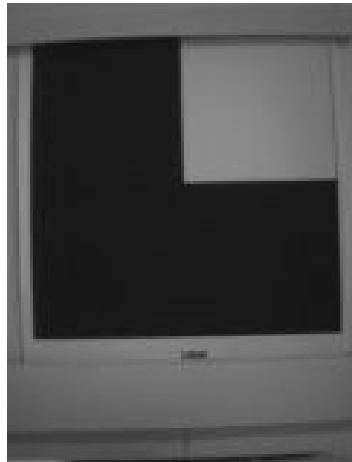
รูปที่ 4.7 รหัสเลขฐานสอง 0001



รูปที่ 4.8 รหัสเลขฐานสอง 0010



รูปที่ 4.9 รหัสเลขฐานสอง 0011



รูปที่ 4.10 รหัสเลขฐานสอง 0100



รูปที่ 4.11 รหัสเลขฐานสอง 0101



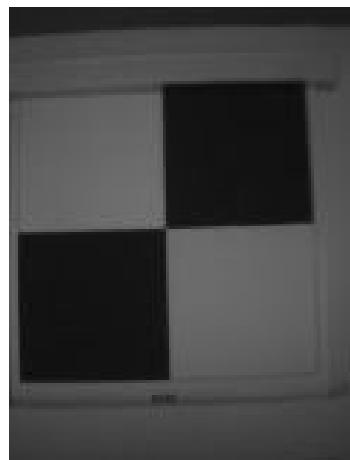
รูปที่ 4.12 รหัสเลขฐานสอง 0110



รูปที่ 4.13 รหัสเลขฐานสอง 0111



รูปที่ 4.14 รหัสเลขฐานสอง 1000



รูปที่ 4.15 รหัสเลขฐานสอง 1001



รูปที่ 4.16 รหัสเลขฐานสอง 1010



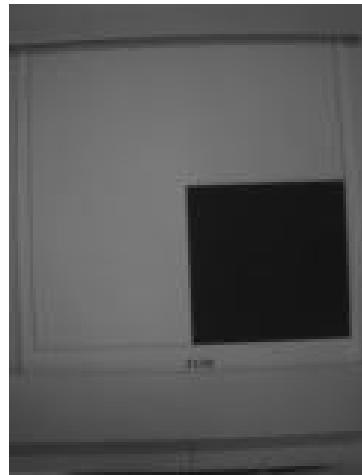
รูปที่ 4.17 รหัสเลขฐานสอง 1011



รูปที่ 4.18 รหัสเลขฐานสอง 1100



รูปที่ 4.19 รหัสเลขฐานสอง 1101



รูปที่ 4.20 รหัสเลขฐานสอง 1110



รูปที่ 4.21 รหัสเลขฐานสอง 1111

ผลลัพธ์การประมวลผลภาพด้วย Machine Learning (ML)

```
In [7]: model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 158, 118, 16)	448
max_pooling2d (MaxPooling2D)	(None, 79, 59, 16)	0
conv2d_1 (Conv2D)	(None, 77, 57, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 38, 28, 32)	0
conv2d_2 (Conv2D)	(None, 36, 26, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 18, 13, 64)	0
flatten (Flatten)	(None, 14976)	0
dropout (Dropout)	(None, 14976)	0
dense (Dense)	(None, 512)	7668224
dense_1 (Dense)	(None, 16)	8208
<hr/>		
Total params: 7,700,016		
Trainable params: 7,700,016		
Non-trainable params: 0		

ตารางที่ 4.1 แสดง layer และ output ของ model

	loss	accuracy	val_loss	val_accuracy
0	3.005446	0.111111	2.747204	0.213198
1	2.916218	0.111111	2.729897	0.111675
2	2.680891	0.111111	2.724812	0.111675
3	2.624065	0.111111	2.724444	0.248731
4	2.611015	0.333333	2.780373	0.228426
5	2.884505	0.111111	2.680954	0.228426
6	2.742844	0.222222	2.620962	0.248731
7	2.676402	0.222222	2.601065	0.233503
8	2.445602	0.222222	2.556686	0.131980
9	2.495266	0.111111	2.474827	0.167513
10	2.400327	0.333333	2.349714	0.197970
11	2.403531	0.000000	2.224885	0.253807
12	2.096408	0.444444	2.179241	0.329949
13	2.072122	0.444444	2.003641	0.319797
14	2.234819	0.111111	1.835598	0.593909
15	1.565629	0.777778	1.610538	0.639594
16	1.038290	0.777778	1.616276	0.532995
17	1.807012	0.555556	1.252061	0.690355
18	1.590370	0.555556	0.903210	0.817259
19	0.753366	0.666667	0.813274	0.776650
20	0.542664	0.888889	0.747176	0.812183
21	1.086576	0.777778	0.531899	0.898477
22	0.225572	1.000000	0.458050	0.918782
23	0.359230	1.000000	0.378730	0.903553
24	0.298625	0.888889	0.343415	0.883249

ตารางที่ 4.2 แสดงค่า loss, accuracy, val_loss, val_accuracy ของการ train แต่ละรอบ

```
In [15]: model.evaluate(validation_dataset)
66/66 [=====] - 0s 6ms/step - loss: 0.3434 - accuracy: 0.8832
Out[15]: [0.3434145450592041, 0.8832487463951111]
```

รูปที่ 4.22 แสดงผลสรุปความถูกต้องเมื่อนำ validation data ทดสอบใน model

Terminal ของ PC1

เมื่อสั่ง capture (หมุนถ่ายทั้ง 3 มุม)

*** Answer from PC2 ***

-45 Degree: 0001

0 Degree: 0010

45 Degree: 0011

*** ----- ***

เมื่อสั่งหมุนไปยังมุมต่าง ๆ ที่ภาพอยู่ จะได้รับผลลัพธ์ดังนี้

*** Answer from PC2 ***

(40,20): 49

(44,24): 50

(48,28): 51

(52,32): 52

mean of QUADRANT 0: 53

(40,70): 97

(44,74): 98

(48,78): 99

(52,82): 100

mean of QUADRANT 1: 101

(100,20): 102

(104,24): 49

(108,28): 50

(112,32): 51

mean of QUADRANT 2: 52

(100,70): 53

(104,74): 49

(108,78): 50

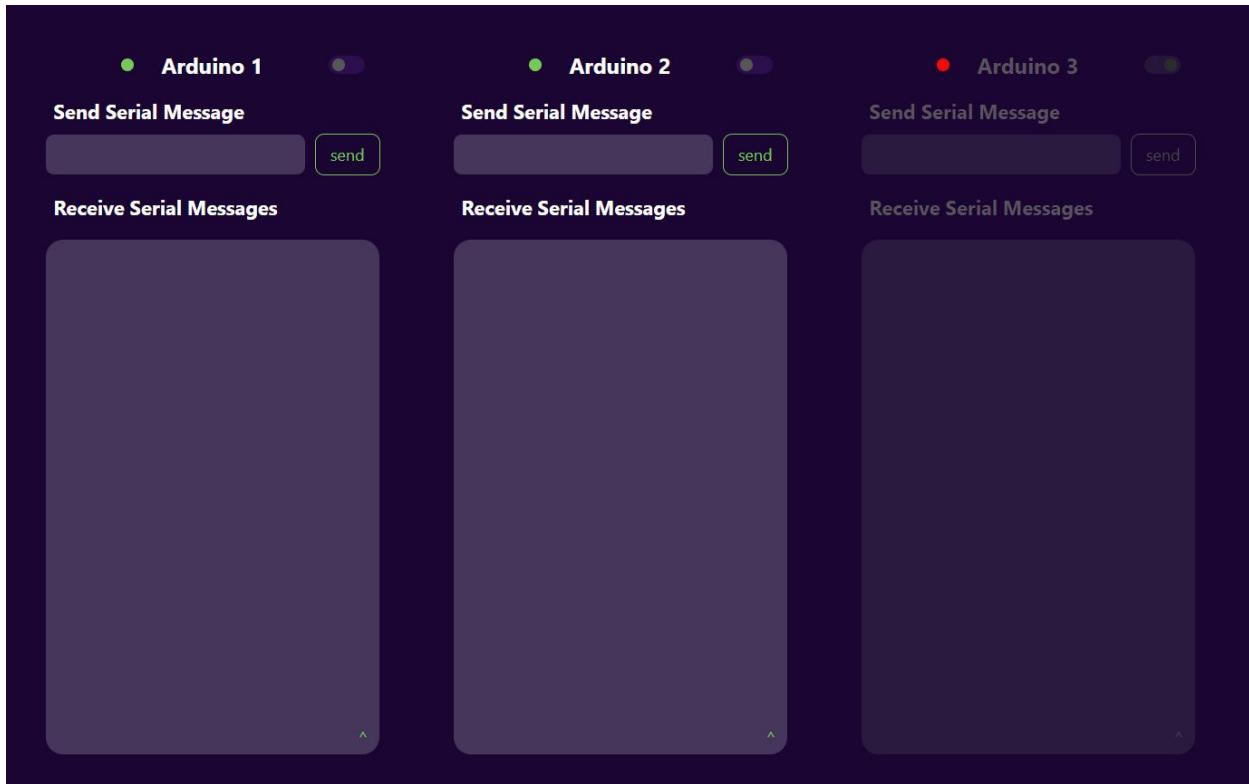
(112,82): 51

mean of QUADRANT 3: 50

*** ----- ***

หน้าเว็บไซต์ (http://ip_server2:5000/)

ใช้สำหรับควบคุม Serial monitor ของ arduino ทั้งสองตัว (ไม่ควบคุม arduino 3 เพราะว่า arduino 3 ทำหน้าที่เพียงถ่ายรูปเท่านั้น)



รูปที่ 4.22 แสดงหน้าเว็บไซต์ http://ip_server2:5000/

บทที่ 5

สรุปผลการดำเนินงาน

สรุปผลการดำเนินงาน

ระบบสามารถสั่งการเริ่มต้นทำงานจาก PC1 ไปสั่งให้ PC2 ทำงานได้โดยใช้คำสั่ง “capture” แล้ว PC2 จะสั่งให้ Arduino ทำการหมุน servo motor ไปตามมุมต่าง ๆ ได้แก่ -45, 0, 45 องศาและทำการเก็บรูปทั้ง 3 มุมแล้วแปลงภาพเป็นเลขฐาน 2 และจึงส่งข้อมูลภาพคืนให้กับ PC1 ทีละรูปโดยแปลงเลขฐาน 2 เป็นตัวอักษรแล้วค่อยแปลงกลับเป็นเลขฐาน 2 ที่ PC1 ต่อมา PC1 จะสามารถเลือกให้ถ่ายรูปใหม่โดยสั่ง “capture” หรือเลือกรูปภาพโดยการพิมพ์เลขฐาน 2 ที่ต้องการเข้าไป หากเลขฐาน 2 นั้นไม่มีอยู่ในรายการภาพที่ถ่ายได้ ก็จะไม่มีการหมุนถ่ายภาพนั้น แต่ถ้ามีอยู่ในรายการที่ถ่ายไว้ได้ในตอนแรกก็จะทำการหมุนไปยังมุมที่ภาพนั้น ๆ อยู่ ทำการถ่ายภาพนั้นใหม่แล้วส่งข้อมูลกลับออกมาเป็นตัวเลขจำนวน 20 ค่าซึ่งนำมารวบค่า 5 ค่าบนแต่ละ quadrant และส่งคืนกลับมาที่ PC1 เพื่อแสดงผล นับเป็นการจบการทำงาน 1 รอบ

ปัญหาอุปสรรคและแนวทางแก้ไข

1. สัญญาณ FM ที่รับได้มี noise มา

แนวทางแก้ไข : ในขั้นต้นได้แก้ปัญหาโดยการทำ Band Pass Filter เพื่อตัด noise ความถี่สูง และต่ำออกไป แต่เมื่อทำการวัดค่าสัญญาณด้วย Oscilloscope แล้วก็ยังพบว่ามี noise อยู่และไม่สามารถรับค่าที่ถูกต้องได้ กลุ่มจึงเห็นสมควรที่ทำการส่งสัญญาณผ่านสายแทน

2. การทำ Parity Bit ใช้เวลาในการทำงานมากเกินไป เพราะต้องวนทำซ้ำ 8 ครั้งต่อข้อมูล 1 ไป

แนวทางแก้ไข : แก้ปัญหาโดยการใช้วิธี checksum ข้อมูลและนำมา mod 2 แทน เพราะแทนที่ได้ด้วยข้อมูล 0 หรือ 1 และมีความสามารถในการเช็คความถูกต้องเท่ากันคือ 50%

3. USB to Serial chip ของ Raspberry Pi มีแค่ 1 ตัวซึ่งทำให้ไม่สามารถทำงานกับหลายอุปกรณ์ได้อย่างเหมาะสม

แนวทางแก้ไข : แก้ปัญหาโดยการสลับเปิด-ปิดอุปกรณ์ที่ต้องสื่อสารผ่าน Serial เมื่อต้องการใช้งานอุปกรณ์ใดก็เปิดใช้อุปกรณ์นั้นและปิดอุปกรณ์ที่เหลือทั้งหมด

4. การเปิด-ปิด Serial ทำให้บอร์ด Arduino รีเซ็ตตัวเอง

แนวทางแก้ไข : แก้ปัญหาโดยการต่อตัวเก็บประจุให้ขาแอนด์เข้ากับขา pin Reset และต่อขาแคริโอดเข้ากับขา GND ของบอร์ด Arduino ทั้งนี้เมื่อต้องการอัพโหลดโคดลงบอร์ด Arduino ต้องทำการนำตัวเก็บประจุออกก่อนอัพโหลด

5. Model ในการทำ ML มีความแม่นยำต่ำ

แนวทางแก้ไข : ในขั้นตอนสามารถแก้ปัญหาได้โดยใช้การตรวจสอบ if-else แทน แต่หากต้องการแก้ไขที่ model ต้องแก้ไขที่ data set โดยการเพิ่ม data set หรือการ Image Augmentation โดยมีหลายวิธี เช่น VerticalShift, Horizontal Shift, Shear, Zoom, Vertical Flip, Horizontal Flip, Rotate, Fill Mode เป็นต้น หรืออีกวิธีที่สามารถแก้ปัญหาความแม่นยำต่ำได้คือ การเปลี่ยนรูปแบบ model ซึ่งมีหลายรูปแบบ เช่น Logistic Regression, Naive Bayes Classifier, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machines เป็นต้น

6. แผ่นบัดกรีขา Pin ทั้ง 18 ที่เป็นตัวเชื่อมกลางระหว่างกล้อง OV7670 และ Arduino เมื่อบัดกรีตามวงจรการต่อกล้องแล้วภาพไม่ชัดอย่างที่ต้องการ ภาพแตก และมี noise อยู่เยอะมาก

แนวทางแก้ไข : ในเบื้องต้นได้ลองใช้ Logic-Probe ในการวัดขาสัญญาณหลังจากสัญญาณที่ผ่านบอร์ดบัดกรี จึงพบว่า ขา V-Sync มีปัญหา เนื่องจากขาสัญญาณนี้ปกติจะเป็น Low และเป็น High เมื่อข้อมูลถูกส่งเรียบร้อยแล้ว ซึ่งข้อมูลนี้วิ่งด้วยความเร็วกว่า 8 ล้านครั้งต่อวินาที (อิงจากโคลด์การส่งข้อมูลจากกล้องถึง Arduino) และใช้สายไฟบัดกรีธรรมด้า จึงทำให้เกิด Crosstalk ไปรบกวนสัญญาณของเส้น V-Sync ทำให้ภาพที่ได้นั้น มี noise และไม่ชัดอย่างที่ต้องการ วิธีการแก้คือ นำตัวเก็บประจุขนาดเล็กมากๆ (ในที่นี้ใช้เป็น 0.750 pF) ต่อที่ขา V-Sync และ ground ของบอร์ด Arduino ทำให้สามารถลด noise และคลื่นรบกวน และทำให้ภาพกลับมาชัดได้เหมือนเดิมได้

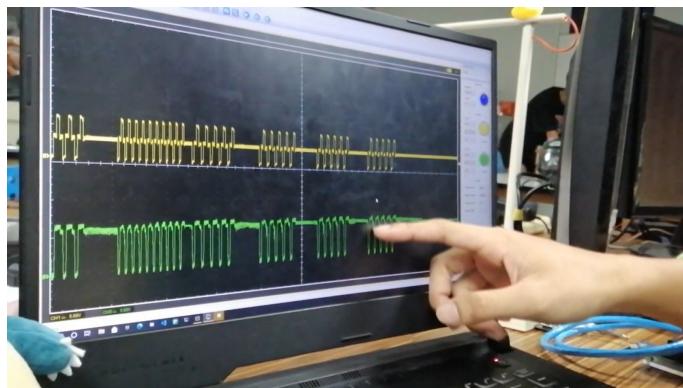
บทที่ 6

สิ่งที่ทำได้เพิ่มเติมหลังการทดลอง

ในการทดลองนี้ เนื่องจากสัญญาณมีคลื่นการรบกวนอยู่เป็นระยะๆ ในช่วงเช้าทั้งวันของการทดลอง จึงทำให้ไม่สามารถส่งผ่านคลื่น FM ได้อย่างมีประสิทธิภาพ ทางผู้จัดทำจึงเห็นสมควรว่า ควรลดTHONมาเป็นการส่งสายเทน เพื่อทำให้ยังคงสภาพการทำงานทั้งหมดให้ยังสามารถดำเนินต่อไปได้อย่างปกติ เมื่อส่งเสร็จ ทางผู้จัดทำได้ทำการเร่งแก้ไขในส่วนต่างๆ เช่นปรับอุปกรณ์ให้ดีขึ้น กำหนด band pass filter ให้ดีขึ้น และใช้ช่วงเวลาที่ไม่มีคลื่นรบกวนอยู่มากเป็นช่วงเวลากลางคืน พบร่วงสามารถส่ง FM ได้เป็นอย่างดี

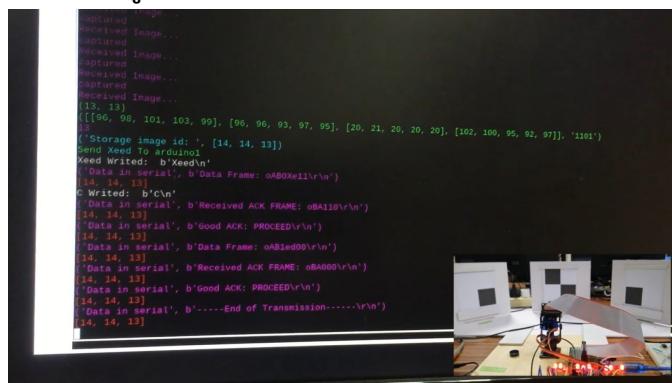
1. สามารถส่งผ่านคลื่น FM ได้แทนการใช้สาย

1.1. อธิบายหลักการทำงาน FM



รูปที่ 6.1 สามารถดูในคลิป VDOdemo_Group1.mp4 ได้ที่วินาทีที่ 7:55 - 10:55 นาที

1.2. Demo การส่งข้อมูลด้วยคลื่น FM



รูปที่ 6.2 สามารถดูในคลิป VDOdemo_Group1.mp4 ได้ที่วินาทีที่ 11:25 - 13:50 นาที

บรรณานุกรม

Forouzan, Behrouz. (2549). **การสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์** แปลและเรียบเรียงจากเรื่อง Data Communications and Networking โดย จักริช พฤษการ. พิมพ์ครั้งที่ 1. กรุงเทพฯ : สำนักพิมพ์ห้อป.

Jirasak Sittigorn. (2563). **Data Communication Assignment 2563**. [ออนไลน์]. เข้าถึงได้จาก : <https://docs.google.com/document/d/1lCAKcCvDOUd78tYZG8W5tjDkQTTrvopiaaOMNXXE0I/edit> (วันที่ค้นข้อมูล : 5 พฤศจิกายน 2563).

Jirasak Sittigorn. (2563). **ขั้นตอนการใช้งาน OV7670 (ไมค์กล้อง)**. [ออนไลน์]. เข้าถึงได้จาก : <https://docs.google.com/document/d/1kdxcITAXJckRb7b2CWud6TY4N25vYgkQhOkC9wr5MgM/edit>. (วันที่ค้นข้อมูล : 5 พฤศจิกายน 2563).

Opening serial ports. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://pyserial.readthedocs.io/en/latest/shortintro.html>. (วันที่ค้นข้อมูล : 7 พฤศจิกายน 2563).

Getting Started with Images. (2563). [ออนไลน์]. เข้าถึงได้จาก : https://opencv-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html. (วันที่ค้นข้อมูล : 8 พฤศจิกายน 2563).

Python Imaging Library. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://pillow.readthedocs.io/en/stable/>. (วันที่ค้นข้อมูล : 8 พฤศจิกายน 2563).

Crosstalk. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://www.sciencedirect.com/topics/engineering/crosstalk>. (วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

Crosstalk Problem. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://axotron.se/blog/crosstalk-problems-when-running-i2c-signals-in-a-cable>. (วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

การแทรกสัญญาณข้าม. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://th.wikipedia.org/wiki/การแทรกสัญญาณข้าม>. (วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

ตัวกลางการเชื่อมต่อเครือข่าย. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://sites.google.com/site/it39000009/hnwy-thi-4?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>. (วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

ในโทรศัมนาคม Crosstalk คืออะไร? (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://www.netinbag.com/th/technology/in-telecommunications-what-is-crosstalk.html>. (วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

Arden Dertat. (2563). **Applied Deep Learning.** [ออนไลน์]. เข้าถึงได้จาก : <https://towardsdatascience.com/applied-deep-learning-part-4-convolutonal-neural-networks-584bc134c1e2>. (วันที่ค้นข้อมูล : 10 พฤศจิกายน 2563).

Natthawat Phongchit. (2563). **Convolutional Neural Networks.** [ออนไลน์]. เข้าถึงได้จาก : <https://medium.com/@natthawatphongchit/มาลองดูวิธีการคิดของ-cnn-กัน-e3f5d73eebaa>. (วันที่ค้นข้อมูล : 10 พฤศจิกายน 2563).

Adit Deshpande. (2563). **Understanding Convolutional Neural Networks.** [ออนไลน์]. เข้าถึงได้จาก : <https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks>. (วันที่ค้นข้อมูล : 11 พฤศจิกายน 2563).

Helix Antenna Design. (2563). [ออนไลน์]. เข้าถึงได้จาก : https://www.changpuak.ch/electronics/calc_12a.php. (วันที่ค้นข้อมูล : 20 พฤศจิกายน 2563).

รูปแบบรายงาน เอกสาร PDF. (2563). [ออนไลน์]. เข้าถึงได้จาก : <https://teemtaro.files.wordpress.com/2013/01/e0b8a3e0b8b9e0b89be0b981e0b89ae0b89ae0b8a3e0b8b2e0b8a2e0b887e0b8b2e0b899.pdf>. (วันที่ค้นข้อมูล : 23 พฤศจิกายน 2563).

ภาคผนวก

โค้ดที่ใช้ในการทำงาน

Protocal.ino

```
#include <Adafruit_MCP4725.h>
#include <Wire.h>
#include "protocol_lib.h"
ProtocolControl* protocol;
void setup()
{
    Serial.begin(115200);
    protocol = new ProtocolControl("A", "B", 90.0);
}

void loop()
{
    protocol->w_wrapper();
}
```

```

protocol.lib.h
#ifndef PROTOCOL_LIB_H
#define PROTOCOL_LIB_H
#include <Arduino.h>
#include "FM_RX.h"
#include "FM_TX.h"
class ProtocolControl
{
public:
    ProtocolControl(String srcName, String destName, float freq);
    ~ProtocolControl();
    String makeDataFrame(String textData, String frameNo, String ENDFLAG, String destName);
    bool approveDataFrame(String);
    String makeAckFrame(String ackNo, String ENDFLAG, String destName);
    bool approveAckFrame(String);
    void wrapper();
    void transmitter();
    void receiver();
    void w_wrapper();
    void w_transmitter();
    void w_receiver();

private:
    String srcName, destName;
    String ackNo;
    String allReceiving;
    String STARTFLAG;
    const int TIMEOUT = 2000;
    const int BACKOFF = 7;

    FM_RX *rx;
    FM_TX *tx;
};

#endif PROTOCOL_LIB_H

```

Protocol.lib.cpp

```
#include <Arduino.h>
#include "protocol_lib.h"

ProtocolControl::ProtocolControl(String srcName, String destName, float freq)
{
    this->srcName = srcName;
    this->destName = destName;
    this->STARTFLAG = "o";
    this->allReceiving = "";
    this->ackNo = "0";
    this->rx = new FM_RX(freq);
    this->tx = new FM_TX();
    Serial.println("Init completed");
}

ProtocolControl::~ProtocolControl()
{
}
```

```
String ProtocolControl::makeDataFrame(String textData, String frameNo, String
ENDFLAG, String destName)
{
    String toSend = ""; //Frame will be store here
    //Header
    toSend += STARTFLAG;
    toSend += destName;
    toSend += srcName;
    toSend += frameNo;

    //Body
    String data = textData;
    while (data.length() < 2) //Add padding: DIFF FROM OLD CODE
    {
        data += "~"; //type with ASCII 126
    }
    toSend += data;
```

```

int sum = 0;
for (int i = 0; i < toSend.length(); i++)
{
    sum += int(toSend[i]);
}
sum = sum % 2;
toSend += String(sum);

toSend += ENDFLAG;
return toSend;

}

bool ProtocolControl::approveDataFrame(String frame) //TODO: CHANGE TO CRC
{
    if (frame.length() != 8)
    {
        Serial.println("Bad Size: " + frame.length());
        return false;
    }

    if (frame[1] != this->srcName[0])
    {
        Serial.println("Not for me: " + String(frame[1]));
        return false;
    }

    if (frame[3] != this->ackNo[0])
    {
        Serial.println("Old Frame: " + String(frame[3]));
        return false;
    }
}

```

```

int sum = 0;
for (int i = 0; i < 6; i++)
{
    sum += int(frame[i]);
}
sum = sum % 2;
char x;
sum == 1 ? x = '1' : x = '0';

if (x == frame[6])
{
    return true;
}
else
{
    return false;
}

}

String ProtocolControl::makeAckFrame(String ackNo, String ENDFLAG, String destName)
{
    String toSend = ""; //Frame will be store here
    //Header
    toSend += STARTFLAG;
    toSend += destName;
    toSend += srcName;

    //Ack Number
    toSend += ackNo;

    int sum = 0;
    for (int i = 0; i < toSend.length(); i++)
    {
        sum += int(toSend[i]);
    }
}

```

```

sum = sum % 2;
toSend += String(sum);

toSend += ENDFLAG;

return toSend;
}

bool ProtocolControl::approveAckFrame(String frame) //TODO: CHANGE TO CRC
{

if (frame.length() != 6)
{
    return false;
}

if (frame[1] != this->srcName[0])
{
    return false;
}

int sum = 0;
for (int i = 0; i < 4; i++)
{
    sum += int(frame[i]);
}
sum = sum % 2;
char x;
sum == 1 ? x = '1' : x = '0';

if (x == frame[4])
{
    return true;
}
else

```

```

{
    return false;
}
}

void ProtocolControl::wrapper()
{
    this->transmitter();
    this->receiver();
}

void ProtocolControl::transmitter()
{
    if (Serial.available()) //Read data from serial
    {
        String frameNo = "0";
        String textData = "";
        const long TIMEOUT = 1000;

        this->ackNo = "0";           //reset ackNo
        this->allReceiving = "";    //reset receiver
        textData = Serial.readStringUntil('\n'); //read data from serial
        Serial.println(textData);

        while (textData.length() > 0) //Send All Data. Frame by Frame
        {
            //Make Data Frame
            String frame = "";
            if (textData.length() > 2)
                frame = this->makeDataFrame(textData.substring(0, 2), frameNo, "1",
this->destName);
            else
                frame = this->makeDataFrame(textData.substring(0, 2), frameNo, "0",
this->destName);
            Serial.println("Data Frame: " + String(frame));
        }
    }
}

```

```

while (true) //Send & "Wait"
{
    for (int i = 0; i < frame.length(); i++) //Send
    {
        this->tx->sendFM(frame[i]);
    }
    long timer = millis();
    bool okAck = false;

    while (millis() - timer < TIMEOUT) //Wait for ACK
    {
        String ackFrame = this->rx->receiveStringFM(6); //receive ACK
        Serial.println("Received ACK FRAME: " + String(ackFrame));
        if (this->approveAckFrame(ackFrame)) //Prep to exit the "Wait"
        {
            Serial.println("Good ACK: PROCEED");
            frameNo == "0" ? frameNo = "1" : frameNo = "0"; //change frame number
            textData = textData.substring(2);
            okAck = true;
            break;
        }
    }

    if (okAck) //Exit "Wait" part
    {
        break;
    }
}

Serial.println("-----End of Transmission-----");
}
}

void ProtocolControl::receiver()
{

```

```

String frame = this->rx->receiveStringFM(8);

if (!frame.equals(""))
{
    Serial.println("Get Frame: " + String(frame));
    if (this->approveDataFrame(frame))
    {
        Serial.println("Good Frame: " + String(frame));
        this->ackNo == "0" ? this->ackNo = "1" : this->ackNo = "0"; //Change Ack Number

        this->allReceiving += frame.substring(4, 6); //Store Incoming Data

        String ackFrame = this->makeAckFrame(ackNo, "0", destName);
        Serial.println("ACK FRAME: " + String(ackFrame));
        for (int i = 0; i < ackFrame.length(); i++)
        {
            this->tx->sendFM(ackFrame[i]);
        }

        if (frame[7] == '0') //End Of Transmission
        {
            this->ackNo = "0";
            //STORE DATA HERE
            Serial.println("----- All Receiving: " + String(this->allReceiving) + " -----");
            this->allReceiving = "";
        }
    }
}

//USE sendFM_noDelay();
void ProtocolControl::w_wrapper()
{
    this->w_transmitter();
    this->w_receiver();
}

```

```

void ProtocolControl::w_transmitter()
{
    if (Serial.available()) //Read data from serial
    {
        String frameNo = "0";
        String textData = "";
        const long TIMEOUT = 200;

        this->ackNo = "0";           //reset ackNo
        this->allReceiving = "";    //reset receiver
        textData = Serial.readStringUntil('\n'); //read data from serial
        //textData = inp; //read data from serial
        //Serial.println(textData);

        while (textData.length() > 0) //Send All Data. Frame by Frame
        {
            //Make Data Frame
            String frame = "";
            if (textData.length() > 2)
                frame = this->makeDataFrame(textData.substring(0, 2), frameNo, "1",
this->destName);
            else
                frame = this->makeDataFrame(textData.substring(0, 2), frameNo, "0",
this->destName);
            Serial.println("Data Frame: " + String(frame));

            while (true) //Send & "Wait"
            {
                for (int i = 0; i < frame.length(); i++) //Send
                {
                    this->tx->sendFM_noDelay(frame[i]);
                }
                long timer = millis();
                bool okAck = false;

```

```

        while (millis() - timer < TIMEOUT) //Wait for ACK
    {
        String ackFrame = this->rx->receiveStringFM(6); //receive ACK
        Serial.println("Received ACK FRAME: " + String(ackFrame));
        if (this->approveAckFrame(ackFrame)) //Prep to exit the "Wait"
        {
            Serial.println("Good ACK: PROCEED");
            frameNo == "0" ? frameNo = "1" : frameNo = "0"; //change frame number
            textData = textData.substring(2);
            okAck = true;
            break;
        }
    }

    if (okAck) //Exit "Wait" part
    {
        break;
    }
}

Serial.println("-----End of Transmission-----");
}
}

void ProtocolControl::w_receiver()
{
    String frame = this->rx->receiveStringFM(8);
    if (!frame.equals(""))
    {
        Serial.println("Get Frame: " + String(frame) + " " + String(this->ackNo));
        if (this->approveDataFrame(frame))
        {
            Serial.println("Good Frame: " + String(frame));
            this->ackNo == "0" ? this->ackNo = "1" : this->ackNo = "0"; //Change Ack Number

            for (int i = 4; i < 6; i++) //store incoming data
            {

```

```

if (frame[i] != '~')
    this->allReceiving += frame[i];
}

String ackFrame = this->makeAckFrame(ackNo, "0", destName);
Serial.println("ACK FRAME: " + String(ackFrame));
for (int i = 0; i < ackFrame.length(); i++)
{
    this->tx->sendFM_noDelay(ackFrame[i]);
}

if (frame[7] == '0') //End Of Transmission
{
    this->ackNo = "0";
    //STORE DATA HERE
    Serial.println("---- All Receiving: " + String(this->allReceiving) + " ----");
    this->allReceiving = "";
}
else {
    String ackFrame = this->makeAckFrame(ackNo, "0", destName);
    Serial.println("ACK FRAME: " + String(ackFrame));
    for (int i = 0; i < ackFrame.length(); i++)
    {
        this->tx->sendFM_noDelay(ackFrame[i]);
    }
}
}
}
}

```

```

FM_TX.h
#include <Adafruit_MCP4725.h>

#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_MCP4725.h>

#define NUM_SAMPLE 4
#define NUM_FREQ 2
#define FREQ_DIFF 100
#define DEF_FREQ 2500

class FM_TX
{
public:
    FM_TX();
    void sendFM(char data);

    void sendFM_noDelay(char data);

private:
    int delay0;
    uint16_t S[NUM_SAMPLE];
    uint16_t S_DAC[NUM_SAMPLE];
    uint16_t freq[NUM_FREQ];
    uint16_t freqDelay[NUM_FREQ];

    Adafruit_MCP4725 dac;

    void setVoltage(uint16_t vol);
    void transmit(char in);
};

```

FM_TX.cpp

```
#include "FM_TX.h"

FM_TX::FM_TX()
{
    Wire.begin();
    dac.begin(0x64);
    for (int i = 0; i < NUM_FREQ; i++)
    {
        freq[i] = ((i + 1) * 5) * FREQ_DIFF;
        freqDelay[i] = ( 1000000 / freq[i] ) / NUM_SAMPLE ;
        Serial.print(freq[i]);
        Serial.print(" ");
        Serial.println(freqDelay[i]);
    }

    for (int i = 0; i < NUM_SAMPLE; i++)
    {
        S[i] = sin(DEG_TO_RAD * 360.0 / NUM_SAMPLE * i);
        S_DAC[i] = S[i] * 2047.5 + 2047.5;
        Serial.print(S[i]);
        Serial.print(" ");
        Serial.println(S_DAC[i]);
    }

    setVoltage(2047);
}

void FM_TX::setVoltage(uint16_t vol)
{
    dac.setVoltage(vol, false);
}

void FM_TX::sendFM(char in)
{
    transmit(in);
```

```

    setVoltage(2047);
    delay(30);
}

void FM_TX::transmit(char in)
{
    int input[8];
    for( int i = 0; i < 8; i++)
    {
        input[i] = (in >> i) & B0001;
    }

    for (int k = 0; k < 8; k++)
    {
        for (int cycle = freq[input[k]] / FREQ_DIFF; cycle > 0; cycle--)
        {
            for (int sample = NUM_SAMPLE - 1; sample >= 0; sample--)
            {
                setVoltage(S_DAC[sample]);
                delayMicroseconds(freqDelay[input[k]]);
            }
        }
    }
}

void FM_TX::sendFM_noDelay(char data)
{
    transmit(data);
    setVoltage(2047);
}

```

```
FM_RX.h
#include <Arduino.h>
#include <Wire.h>
#include "TEA5767.h"

#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

class FM_RX
{
public:
    FM_RX(float freq);
    int receiveFM();
    String receiveStringFM(int maxLength);

private:
    TEA5767* radio;

    int8_t isPeek(uint16_t);
};
```

FM_RX.cpp

```
#include "FM_RX.h"
```

```
FM_RX::FM_RX(float freq)
{
    // put your setup code here, to run once:
    sbi(ADCSRA, ADPS2);
    cbi(ADCSRA, ADPS1);
    cbi(ADCSRA, ADPS0);

    Wire.begin();
    radio = new TEA5767(freq);
    radio->init();
    radio->set_frequency(freq);
}
```

```
int FM_RX::receiveFM()
{
    int prev = 0;
    int count = 0;

    uint16_t data = 0;
    uint16_t bit_check = 0;

    bool check_baud = false;
    bool check_amp = false;

    uint32_t baud_begin = micros();

    while (micros() - baud_begin < 40000)
    {
        int tmp = isPeek(analogRead(A2));

        if (tmp == 1 and prev == 0 and !check_amp) // check amplitude
        {
            check_amp = true; // is first max amplitude in that baud
        }
    }
}
```

```

if ( !check_baud )
{
    baud_begin = micros();
}
}

if (tmp == 0 and check_baud) {
    if (micros() - baud_begin > 9800 )
    {
        int dt = ((int(floor((count) / 5.0) - 1)) & 1);
        uint16_t last = dt << (bit_check);
        data |= last;

        bit_check++;
        if (bit_check == 8) // 8 bits
        {
            if ( data != 0)
                return data;
            data = 0;
            bit_check = 0;
        }
        check_baud = false;
        count = 0;
    }
}

if (tmp == 0 and prev == 1 and check_amp) {
    count++;
    check_baud = true;
    check_amp = false;
}
prev = tmp;
}

return -1;
}

```

```

int8_t FM_RX::isPeek(uint16_t val)
{
    if (val <= 200)
        //if (val >= 430)//Using 600 with amplifier
        return 1;
    else
        return 0;
}

String FM_RX::receiveStringFM(int maxLength)//Return data string (Empty string if
nothing arrive)
{
    int prev = 0;
    int count = 0;

    uint16_t data = 0;
    uint16_t bit_check = 0;

    bool check_baud = false;
    bool check_amp = false;

    uint32_t baud_begin = micros();

    String message = "";

    while (micros() - baud_begin < 40000)
    {
        int tmp = isPeek(analogRead(A2));

        if (tmp == 1 and prev == 0 and !check_amp) // check amplitude
        {
            check_amp = true; // is first max amplitude in that baud
            if (!check_baud)
            {
                baud_begin = micros();

```

```

        }

    }

if (tmp == 0 and check_baud) {
    if (micros() - baud_begin > 9800 )
    {

        int dt = ((int(floor((count) / 5.0)) - 1) & 1);
        uint16_t last = dt << (bit_check);
        data |= last;

        bit_check++;
        if (bit_check == 8) // 8 bits
        {
            if (data != 0)
                message += char(data);
            if (message.length() == maxLength)
                break;
            data = 0;
            bit_check = 0;
        }
        check_baud = false;
        count = 0;
    }
}

if (tmp == 0 and prev == 1 and check_amp) {
    count++;
    check_baud = true;
    check_amp = false;
}
prev = tmp;
}

return message;
}

```

ServoController.h

```
#ifndef MY_LIBRARY_H
#define MY_LIBRARY_H
#include <Servo.h>
#include <EEPROM.h>
#include <Arduino.h>
void init_serve(int pin_horizontal_Servo, int pin_vertical_Servo);
int readValue(int address);
void writeValue(int address, int val);
void saveValue(char degree, int h, int t);
bool isCommandToServo(String command);
int valueFromString(char x0, char x1, char x2);
void moveTo(char degree);
void writeServoTo(int ser, int newPos);
String getSettingAll();
#endif
```

ServoController.cpp

```
#include "ServoController.h"
#define R_ADDRESS 0
#define R_T_ADDRESS 1
#define M_ADDRESS 2
#define M_T_ADDRESS 3
#define L_ADDRESS 4
#define L_T_ADDRESS 5
int R_DEGREE = 0;
int R_T_DEGREE = 0;
int M_DEGREE = 0;
int M_T_DEGREE = 0;
int L_DEGREE = 0;
int L_T_DEGREE = 0;
Servo vertical_Servo;
Servo horizontal_Servo;

void writeServoTo(int ser, int newPos) {
    /*
        ser
        0 = horizontal_Servo
        1 = vertical_Servo
    */
    int now_degree = 0;
    if (ser == 0) {
        now_degree = horizontal_Servo.read();
        if (now_degree < newPos) {
            for (int i = now_degree ; i < newPos ; i++ ) {
                horizontal_Servo.write(i);
                delay(5);
            }
        }
    } else {
        for (int i = now_degree ; i > newPos ; i-- ) {
            horizontal_Servo.write(i);
            delay(5);
        }
    }
}
```

```

        }
    }
}

else if (ser == 1) {
    now_degree = vertical_Servo.read();
    if (now_degree < newPos) {
        for (int i = now_degree ; i < newPos ; i++ ) {
            vertical_Servo.write(i);
            delay(5);
        }
    }
    else {
        for (int i = now_degree ; i > newPos ; i-- ) {
            vertical_Servo.write(i);
            delay(5);
        }
    }
}
}

```

```

void moveTo(char degree) {
    writeServoTo(1, 130);
    switch (degree) {
        case 'L':
            writeServoTo(0, L_DEGREE);
            writeServoTo(1, L_T_DEGREE);
            break;
        case 'M':
            writeServoTo(0, M_DEGREE);
            writeServoTo(1, M_T_DEGREE);
            break;
        case 'R':
            writeServoTo(0, R_DEGREE);
            writeServoTo(1, R_T_DEGREE);
    }
}

```

```

        break;
    case 'C':
        horizontal_Servo.write(90);
        vertical_Servo.write(45);
        break;
    }
}

int valueFromString(char x0, char x1, char x2) {
    return (x0 - '0') * 100 + (x1 - '0') * 10 + (x2 - '0') * 1;
}

void updateValue() {
    R_DEGREE = readValue(R_ADDRESS);
    R_T_DEGREE = readValue(R_T_ADDRESS);
    L_DEGREE = readValue(L_ADDRESS);
    L_T_DEGREE = readValue(L_T_ADDRESS);
    M_DEGREE = readValue(M_ADDRESS);
    M_T_DEGREE = readValue(M_T_ADDRESS);
}

bool isCommandToServo(String command) {
    int haveNewLine = command[command.length() - 1] == '\n' ? 0 : 1;
    if (command.length() == 8 - haveNewLine) {
        saveValue(command[0], valueFromString(command[1], command[2], command[3]),
        valueFromString(command[4], command[5], command[6]));
        updateValue();
        return true;
    }
    else if (command.length() == 2 - haveNewLine) {
        moveTo(command[0]);
        return true;
    }
    return false;
}

```

```

void saveValue(char degree, int h, int v) {
    switch (degree) {
        case 'L':
            writeValue(L_ADDRESS, h);
            writeValue(L_T_ADDRESS, v);
            break;
        case 'M':
            writeValue(M_ADDRESS, h);
            writeValue(M_T_ADDRESS, v);
            break;
        case 'R':
            writeValue(R_ADDRESS, h);
            writeValue(R_T_ADDRESS, v);
            break;
    }
}

int readValue(int address) {
    return EEPROM.read(address);
}

void writeValue(int address, int val) {
    EEPROM.write(address, val);
}

void init_serve(int pin_horizontal_Servo, int pin_vertical_Servo) {
    horizontal_Servo.attach(pin_horizontal_Servo);
    vertical_Servo.attach(pin_vertical_Servo);
    horizontal_Servo.write(90);
    vertical_Servo.write(45);
    updateValue();
    Serial.begin(115200);
}

```

Camera.ino

```
#include "CameraController.h"
```

```
void setup() {
    initCamera(0x80 | 3);
}
```

```
void loop() {
    capture(160, 120);
}
```

CameraController.h

```
#ifndef MY_LIBRARY_H
#define MY_LIBRARY_H
#include<Arduino.h>
void error_led(void);
void twiStart(void);
void twiWriteByte(uint8_t DATA, uint8_t type);
void twiAddr(uint8_t addr, uint8_t typeTWI);
void wrReg(uint8_t reg, uint8_t dat);
static uint8_t twiRd(uint8_t nack);
uint8_t rdReg(uint8_t reg);
void wrSensorRegs8_8(const struct regval_list reglist[]);
void setColor(void);
void setRes(void);
void camInit(void);
void arduinoUnoInut(void);
void StringPgm(const char * str);
static void captureImg(uint16_t wg, uint16_t hg);
void initCamera(int reg);
void capture(int width_, int height_);
#endif
```

CameraController.cpp

```
#include "CameraController.h"
#include <stdint.h>
#include <avr/io.h>
#include <util/twi.h>
#include <util/delay.h>
#include <avr/pgmspace.h>

#define F_CPU 16000000UL
#define vga 0
#define qvga 1
#define qqvga 2
#define yuv422 0
#define rgb565 1
#define bayerRGB 2
#define camAddr_WR 0x42
#define camAddr_RD 0x43

/* Registers */
#define REG_GAIN 0x00 /* Gain lower 8 bits (rest in vref) */
#define REG_BLUE 0x01 /* blue gain */
#define REG_RED 0x02 /* red gain */
#define REG_VREF 0x03 /* Pieces of GAIN, VSTART, VSTOP */
#define REG_COM1 0x04 /* Control 1 */
#define COM1_CCIR656 0x40 /* CCIR656 enable */

#define REG_BAVE 0x05 /* U/B Average level */
#define REG_GbAVE 0x06 /* Y/Gb Average level */
#define REG_AECHH 0x07 /* AEC MS 5 bits */
#define REG_RAVE 0x08 /* V/R Average level */
#define REG_COM2 0x09 /* Control 2 */
#define COM2_SSLEEP 0x10 /* Soft sleep mode */
#define REG_PID 0xa /* Product ID MSB */
#define REG_VER 0xb /* Product ID LSB */
#define REG_COM3 0xc /* Control 3 */
#define COM3_SWAP 0x40 /* Byte swap */
```

```

#define COM3_SCALEEN      0x08 /* Enable scaling */
#define COM3_DCWEN        0x04 /* Enable downsample/crop/window */
#define REG_COM4          0x0d /* Control 4 */
#define REG_COM5          0x0e /* All "reserved" */
#define REG_COM6          0x0f /* Control 6 */
#define REG_AECH          0x10 /* More bits of AEC value */
#define REG_CLKRC          0x11 /* Clock control */
#define CLK_EXT            0x40 /* Use external clock directly */
#define CLK_SCALE           0x3f /* Mask for internal clock scale */
#define REG_COM7          0x12 /* Control 7 */ //REG mean address.
#define COM7_RESET          0x80 /* Register reset */
#define COM7_FMT_MASK       0x38
#define COM7_FMT_VGA         0x00
#define COM7_FMT_CIF         0x20 /* CIF format */
#define COM7_FMT_QVGA        0x10 /* QVGA format */
#define COM7_FMT_QCIF        0x08 /* QCIF format */
#define COM7_RGB             0x04 /* bits 0 and 2 - RGB format */
#define COM7_YUV             0x00 /* YUV */
#define COM7_BAYER           0x01 /* Bayer format */
#define COM7_PBAYER          0x05 /* "Processed bayer" */
#define REG_COM8          0x13 /* Control 8 */
#define COM8_FASTAEC         0x80 /* Enable fast AGC/AEC */
#define COM8_AECSTEP          0x40 /* Unlimited AEC step size */
#define COM8_BFILT            0x20 /* Band filter enable */
#define COM8_AGC              0x04 /* Auto gain enable */
#define COM8_AWB              0x02 /* White balance enable */
#define COM8_AEC              0x01 /* Auto exposure enable */
#define REG_COM9          0x14 /* Control 9- gain ceiling */
#define REG_COM10         0x15 /* Control 10 */
#define COM10_HSYNC           0x40 /* HSYNC instead of HREF */
#define COM10_PCLK_HB          0x20 /* Suppress PCLK on horiz blank */
#define COM10_HREF_REV          0x08 /* Reverse HREF */
#define COM10_VS_LEAD           0x04 /* VSYNC on clock leading edge */
#define COM10_VS_NEG            0x02 /* VSYNC negative */
#define COM10_HS_NEG            0x01 /* HSYNC negative */
#define REG_HSTART          0x17 /* Horiz start high bits */

```

```

#define REG_HSTOP 0x18 /* Horiz stop high bits */
#define REG_VSTART 0x19 /* Vert start high bits */
#define REG_VSTOP 0x1a /* Vert stop high bits */
#define REG_PSHFT 0x1b /* Pixel delay after HREF */
#define REG_MIDH 0x1c /* Manuf. ID high */
#define REG_MIDL 0x1d /* Manuf. ID low */
#define REG_MVFP 0x1e /* Mirror / vflip */
#define MVFP_MIRROR 0x20 /* Mirror image */
#define MVFP_FLIP 0x10 /* Vertical flip */

#define REG_AEW 0x24 /* AGC upper limit */
#define REG_AEB 0x25 /* AGC lower limit */
#define REG_VPT 0x26 /* AGC/AEC fast mode op region */
#define REG_HSYST 0x30 /* HSYNC rising edge delay */
#define REG_HSYEN 0x31 /* HSYNC falling edge delay */
#define REG_HREF 0x32 /* HREF pieces */
#define REG_TSLB 0x3a /* lots of stuff */
#define TSLB_YLAST 0x04 /* UYVY or VYUY - see com13 */
#define REG_COM11 0x3b /* Control 11 */
#define COM11_NIGHT 0x80 /* Night mode enable */
#define COM11_NMFR 0x60 /* Two bit NM frame rate */
#define COM11_HZAUTO 0x10 /* Auto detect 50/60 Hz */
#define COM11_50HZ 0x08 /* Manual 50Hz select */
#define COM11_EXP 0x02
#define REG_COM12 0x3c /* Control 12 */
#define COM12_HREF 0x80 /* HREF always */
#define REG_COM13 0x3d /* Control 13 */
#define COM13_GAMMA 0x80 /* Gamma enable */
#define COM13_UVSAT 0x40 /* UV saturation auto adjustment */
#define COM13_UVSWAP 0x01 /* V before U - w/TSLB */
#define REG_COM14 0x3e /* Control 14 */
#define COM14_DCWEN 0x10 /* DCW/PCLK-scale enable */
#define REG_EDGE 0x3f /* Edge enhancement factor */
#define REG_COM15 0x40 /* Control 15 */
#define COM15_R10F0 0x00 /* Data range 10 to F0 */
#define COM15_R01FE 0x80 /* 01 to FE */

```

```

#define COM15_R00FF      0xc0 /* 00 to FF */
#define COM15_RGB565     0x10 /* RGB565 output */
#define COM15_RGB555     0x30 /* RGB555 output */
#define REG_COM16 0x41 /* Control 16 */
#define COM16_AWBGAIN    0x08 /* AWB gain enable */
#define REG_COM17 0x42 /* Control 17 */
#define COM17_AECWIN     0xc0 /* AEC window - must match COM4 */
#define COM17_CBAR        0x08 /* DSP Color bar */
*/

```

This matrix defines how the colors are generated, must be tweaked to adjust hue and saturation.

Order: v-red, v-green, v-blue, u-red, u-green, u-blue
 They are nine-bit signed quantities, with the sign bit stored in 0x58. Sign for v-red is bit 0, and up from there.

```

*/
#define REG_CMATRIX_BASE 0x4f
#define CMATRIX_LEN       6
#define REG_CMATRIX_SIGN  0x58
#define REG_BRIGHT         0x55 /* Brightness */
#define REG_CONTRAS        0x56 /* Contrast control */
#define REG_GFIX           0x69 /* Fix gain control */
#define REG_REG76          0x76 /* OV's name */
#define R76_BLKPCOR        0x80 /* Black pixel correction enable */
#define R76_WHTPCOR        0x40 /* White pixel correction enable */
#define REG_RGB444          0x8c /* RGB 444 control */
#define R444_ENABLE         0x02 /* Turn on RGB444, overrides 5x5 */
#define R444_RGBX           0x01 /* Empty nibble at end */
#define REG_HAECC1          0x9f /* Hist AEC/AGC control 1 */
#define REG_HAECC2          0xa0 /* Hist AEC/AGC control 2 */
#define REG_BD50MAX         0xa5 /* 50hz banding step limit */
#define REG_HAECC3          0xa6 /* Hist AEC/AGC control 3 */
#define REG_HAECC4          0xa7 /* Hist AEC/AGC control 4 */
#define REG_HAECC5          0xa8 /* Hist AEC/AGC control 5 */
#define REG_HAECC6          0xa9 /* Hist AEC/AGC control 6 */
#define REG_HAECC7          0xaa /* Hist AEC/AGC control 7 */

```

```

#define REG_BD60MAX      0xab /* 60hz banding step limit */
#define REG_GAIN     0x00 /* Gain lower 8 bits (rest in vref) */
#define REG_BLUE    0x01 /* blue gain */
#define REG_RED     0x02 /* red gain */
#define REG_VREF    0x03 /* Pieces of GAIN, VSTART, VSTOP */
#define REG_COM1    0x04 /* Control 1 */
#define COM1_CCIR656 0x40 /* CCIR656 enable */
#define REG_BAVE    0x05 /* U/B Average level */
#define REG_GbAVE   0x06 /* Y/Gb Average level */
#define REG_AECHH   0x07 /* AEC MS 5 bits */
#define REG_RAVE    0x08 /* V/R Average level */
#define REG_COM2    0x09 /* Control 2 */
#define COM2_SSLEEP 0x10 /* Soft sleep mode */
#define REG_PID     0xa /* Product ID MSB */
#define REG_VER     0xb /* Product ID LSB */
#define REG_COM3    0xc /* Control 3 */
#define COM3_SWAP   0x40 /* Byte swap */
#define COM3_SCALEEN 0x08 /* Enable scaling */
#define COM3_DCWEN  0x04 /* Enable downsamp/crop/window */
#define REG_COM4    0xd /* Control 4 */
#define REG_COM5    0xe /* All "reserved" */
#define REG_COM6    0xf /* Control 6 */
#define REG_AECH    0x10 /* More bits of AEC value */
#define REG_CLKRC   0x11 /* Clock control */
#define CLK_EXT     0x40 /* Use external clock directly */
#define CLK_SCALE   0x3f /* Mask for internal clock scale */
#define REG_COM7    0x12 /* Control 7 */
#define COM7_RESET  0x80 /* Register reset */
#define COM7_FMT_MASK 0x38
#define COM7_FMT_VGA 0x00
#define COM7_FMT_CIF 0x20 /* CIF format */
#define COM7_FMT_QVGA 0x10 /* QVGA format */
#define COM7_FMT_QCIF 0x08 /* QCIF format */
#define COM7_RGB    0x04 /* bits 0 and 2 - RGB format */
#define COM7_YUV    0x00 /* YUV */
#define COM7_BAYER  0x01 /* Bayer format */

```

```

#define COM7_PBAYER      0x05 /* "Processed bayer" */
#define REG_COM8   0x13 /* Control 8 */
#define COM8_FASTAEC     0x80 /* Enable fast AGC/AEC */
#define COM8_AECSTEP      0x40 /* Unlimited AEC step size */
#define COM8_BFILT       0x20 /* Band filter enable */
#define COM8_AGC        0x04 /* Auto gain enable */
#define COM8_AWB        0x02 /* White balance enable */
#define COM8_AEC        0x01 /* Auto exposure enable */
#define REG_COM9   0x14 /* Control 9- gain ceiling */
#define REG_COM10  0x15 /* Control 10 */
#define COM10_HSYNC      0x40 /* HSYNC instead of HREF */
#define COM10_PCLK_HB    0x20 /* Suppress PCLK on horiz blank */
#define COM10_HREF_REV   0x08 /* Reverse HREF */
#define COM10_VS_LEAD    0x04 /* VSYNC on clock leading edge */
#define COM10_VS_NEG     0x02 /* VSYNC negative */
#define COM10_HS_NEG     0x01 /* HSYNC negative */
#define REG_HSTART      0x17 /* Horiz start high bits */
#define REG_HSTOP       0x18 /* Horiz stop high bits */
#define REG_VSTART      0x19 /* Vert start high bits */
#define REG_VSTOP       0x1a /* Vert stop high bits */
#define REG_PSHFT       0x1b /* Pixel delay after HREF */
#define REG_MIDH        0x1c /* Manuf. ID high */
#define REG_MIDL        0x1d /* Manuf. ID low */
#define REG_MVFP        0x1e /* Mirror / vflip */
#define MVFP_MIRROR     0x20 /* Mirror image */
#define MVFP_FLIP       0x10 /* Vertical flip */
#define REG_AEW         0x24 /* AGC upper limit */
#define REG_AEB         0x25 /* AGC lower limit */
#define REG_VPT         0x26 /* AGC/AEC fast mode op region */
#define REG_HSYST      0x30 /* HSYNC rising edge delay */
#define REG_HSYEN      0x31 /* HSYNC falling edge delay */
#define REG_HREF       0x32 /* HREF pieces */
#define REG_TSLB        0x3a /* lots of stuff */
#define TSLB_YLAST     0x04 /* UYVY or VYUY - see com13 */
#define REG_COM11      0x3b /* Control 11 */
#define COM11_NIGHT     0x80 /* NIght mode enable */

```

```

#define COM11_NMFR      0x60 /* Two bit NM frame rate */
#define COM11_HZAUTO    0x10 /* Auto detect 50/60 Hz */
#define COM11_50HZ       0x08 /* Manual 50Hz select */
#define COM11_EXP        0x02
#define REG_COM12 0x3c /* Control 12 */
#define COM12_HREF      0x80 /* HREF always */
#define REG_COM13 0x3d /* Control 13 */
#define COM13_GAMMA     0x80 /* Gamma enable */
#define COM13_UVSAT      0x40 /* UV saturation auto adjustment */
#define COM13_UVSWAP     0x01 /* V before U - w/TSLB */
#define REG_COM14 0x3e /* Control 14 */
#define COM14_DCWEN     0x10 /* DCW/PCLK-scale enable */
#define REG_EDGE   0x3f /* Edge enhancement factor */
#define REG_COM15 0x40 /* Control 15 */
#define COM15_R10F0      0x00 /* Data range 10 to F0 */
#define COM15_R01FE      0x80 /* 01 to FE */
#define COM15_R00FF      0xc0 /* 00 to FF */
#define COM15_RGB565     0x10 /* RGB565 output */
#define COM15_RGB555     0x30 /* RGB555 output */
#define REG_COM16 0x41 /* Control 16 */
#define COM16_AWBGAIN    0x08 /* AWB gain enable */
#define REG_COM17 0x42 /* Control 17 */
#define COM17_AECWIN     0xc0 /* AEC window - must match COM4 */
#define COM17_CBAR        0x08 /* DSP Color bar */

#define CMATRIX_LEN       6
#define REG_BRIGHT 0x55 /* Brightness */
#define REG_REG76 0x76 /* OV's name */
#define R76_BLKPCOR     0x80 /* Black pixel correction enable */
#define R76_WHTPCOR     0x40 /* White pixel correction enable */
#define REG_RGB444      0x8c /* RGB 444 control */
#define R444_ENABLE      0x02 /* Turn on RGB444, overrides 5x5 */
#define R444_RGBX        0x01 /* Empty nibble at end */
#define REG_HAECC1       0x9f /* Hist AEC/AGC control 1 */
#define REG_HAECC2       0xa0 /* Hist AEC/AGC control 2 */
#define REG_BD50MAX      0xa5 /* 50hz banding step limit */

```

```

#define REG_HAECC3 0xa6 /* Hist AEC/AGC control 3 */
#define REG_HAECC4 0xa7 /* Hist AEC/AGC control 4 */
#define REG_HAECC5 0xa8 /* Hist AEC/AGC control 5 */
#define REG_HAECC6 0xa9 /* Hist AEC/AGC control 6 */
#define REG_HAECC7 0xaa /* Hist AEC/AGC control 7 */
#define REG_BD60MAX 0xab /* 60hz banding step limit */

#define MTX1 0x4f /* Matrix Coefficient 1 */
#define MTX2 0x50 /* Matrix Coefficient 2 */
#define MTX3 0x51 /* Matrix Coefficient 3 */
#define MTX4 0x52 /* Matrix Coefficient 4 */
#define MTX5 0x53 /* Matrix Coefficient 5 */
#define MTX6 0x54 /* Matrix Coefficient 6 */

#define REG_CONTRAS 0x56 /* Contrast control */
#define MTXS 0x58 /* Matrix Coefficient Sign */
#define AWBC7 0x59 /* AWB Control 7 */
#define AWBC8 0x5a /* AWB Control 8 */
#define AWBC9 0x5b /* AWB Control 9 */
#define AWBC10 0x5c /* AWB Control 10 */
#define AWBC11 0x5d /* AWB Control 11 */
#define AWBC12 0x5e /* AWB Control 12 */
#define REG_GFI 0x69 /* Fix gain control */
#define GGAIN 0x6a /* G Channel AWB Gain */
#define DBLV 0x6b

#define AWBCTR3 0x6c /* AWB Control 3 */
#define AWBCTR2 0x6d /* AWB Control 2 */
#define AWBCTR1 0x6e /* AWB Control 1 */
#define AWBCTR0 0x6f /* AWB Control 0 */

struct regval_list {
    uint8_t reg_num;
    uint16_t value;
};

const struct regval_list qvga_ov7670[] PROGMEM = {
    { REG_COM14, 0x19 },

```

```

{ 0x72, 0x11 },
{ 0x73, 0xf1 },

{ REG_HSTART, 0x16 },
{ REG_HSTOP, 0x04 },
{ REG_HREF, 0xa4 },
{ REG_VSTART, 0x02 },
{ REG_VSTOP, 0x7a },
{ REG_VREF, 0x0a },

/* { REG_HSTART, 0x16 },
{ REG_HSTOP, 0x04 },
{ REG_HREF, 0x24 },
{ REG_VSTART, 0x02 },
{ REG_VSTOP, 0x7a },
{ REG_VREF, 0x0a },*/
{ 0xff, 0xff }, /* END MARKER */
};

const struct regval_list qqvga_ov7670[] PROGMEM = {
// {REG_COM3, COM3_DCWEN}, // enable downsample/crop/window

{REG_COM14, 0x1a}, // divide by 4
{0x72, 0x22}, // downsample by 4
{0x73, 0xf2}, // divide by 4
{REG_HSTART, 0x16},
{REG_HSTOP, 0x04},
{REG_HREF, 0xa4},
{REG_VSTART, 0x02},
{REG_VSTOP, 0x7a},
{REG_VREF, 0x0a},
{0xff, 0xff}, /* END MARKER */
};

const struct regval_list yuv422_ov7670[] PROGMEM = {

```

```

{ REG_COM7, 0x0 }, /* Selects YUV mode */
{ REG_RGB444, 0 }, /* No RGB444 please */
{ REG_COM1, 0 },
{ REG_COM15, COM15_R00FF },
{ REG_COM9, 0x6A }, /* 128x gain ceiling; 0x8 is reserved bit */
{ 0x4f, 0x80 }, /* "matrix coefficient 1" */
{ 0x50, 0x80 }, /* "matrix coefficient 2" */
{ 0x51, 0 }, /* vb */
{ 0x52, 0x22 }, /* "matrix coefficient 4" */
{ 0x53, 0x5e }, /* "matrix coefficient 5" */
{ 0x54, 0x80 }, /* "matrix coefficient 6" */
{ REG_COM13, COM13_UVSAT },
{ 0xff, 0xff }, /* END MARKER */
};


```

```

const struct regval_list ov7670_default_regs[] PROGMEM = //from the linux driver
{
    { REG_COM7, COM7_RESET },
    { REG_TSLB, 0x04 }, /* OV */
    { REG_COM7, 0 }, /* VGA */
    { REG_HSTART, 0x13 }, { REG_HSTOP, 0x01 },
    { REG_HREF, 0xb6 }, { REG_VSTART, 0x02 },
    { REG_VSTOP, 0x7a }, { REG_VREF, 0xa },
    { REG_COM3, 0 }, { REG_COM14, 0 },
    { 0x70, 0x3a }, { 0x71, 0x35 },
    { 0x72, 0x11 }, { 0x73, 0xf0 },
    { 0xa2, /* 0x02 changed to 1 */ 1 }, { REG_COM10, 0x0 },
    /* Gamma curve values */
    { 0x7a, 0x20 }, { 0x7b, 0x10 },
    { 0x7c, 0x1e }, { 0x7d, 0x35 },
    { 0x7e, 0x5a }, { 0x7f, 0x69 },
    { 0x80, 0x76 }, { 0x81, 0x80 },
    { 0x82, 0x88 }, { 0x83, 0x8f },
    { 0x84, 0x96 }, { 0x85, 0xa3 },
    { 0x86, 0xaf }, { 0x87, 0xc4 },
};


```

```

{ 0x88, 0xd7 }, { 0x89, 0xe8 },
/* AGC and AEC parameters. Note we start by disabling those features,
then turn them only after tweaking the values.*/
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP },
{ REG_GAIN, 0 }, { REG_AECH, 0 },
{ REG_COM4, 0x40 }, /* magic reserved bit */
{ REG_COM9, 0x18 }, /* 4x gain + magic rsrv bit */
{ REG_BD50MAX, 0x05 }, { REG_BD60MAX, 0x07 },
{ REG_AEW, 0x95 }, { REG_AEB, 0x33 },
{ REG_VPT, 0xe3 }, { REG_HAECC1, 0x78 },
{ REG_HAECC2, 0x68 }, { 0xa1, 0x03 }, /* magic */
{ REG_HAECC3, 0xd8 }, { REG_HAECC4, 0xd8 },
{ REG_HAECC5, 0xf0 }, { REG_HAECC6, 0x90 },
{ REG_HAECC7, 0x94 },
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP | COM8_AGC | COM8_AEC },
{ 0x30, 0 }, { 0x31, 0 }, // disable some delays

{ REG_COM5, 0x61 }, { REG_COM6, 0x4b },
{ 0x16, 0x02 }, { REG_MVFP, 0x07 },
{ 0x21, 0x02 }, { 0x22, 0x91 },
{ 0x29, 0x07 }, { 0x33, 0x0b },
{ 0x35, 0x0b }, { 0x37, 0x1d },
{ 0x38, 0x71 }, { 0x39, 0x2a },
{ REG_COM12, 0x78 }, { 0x4d, 0x40 },
{ 0x4e, 0x20 }, { REG_GFIX, 0 },
/*{0x6b, 0x4a},*/ { 0x74, 0x10 },
{ 0x8d, 0x4f }, { 0x8e, 0 },
{ 0x8f, 0 }, { 0x90, 0 },
{ 0x91, 0 }, { 0x96, 0 },
{ 0x9a, 0 }, { 0xb0, 0x84 },
{ 0xb1, 0x0c }, { 0xb2, 0x0e },
{ 0xb3, 0x82 }, { 0xb8, 0xa },

/* More reserved magic, some of which tweaks white balance */
{ 0x43, 0x0a }, { 0x44, 0xf0 },
{ 0x45, 0x34 }, { 0x46, 0x58 },

```

```

{ 0x47, 0x28 }, { 0x48, 0x3a },
{ 0x59, 0x88 }, { 0x5a, 0x88 },
{ 0x5b, 0x44 }, { 0x5c, 0x67 },
{ 0x5d, 0x49 }, { 0x5e, 0x0e },
{ 0x6c, 0x0a }, { 0x6d, 0x55 },
{ 0x6e, 0x11 }, { 0x6f, 0x9e }, /* it was 0x9F "9e for advance AWB" */
{ 0x6a, 0x40 }, { REG_BLUE, 0x40 },
{ REG_RED, 0x60 },
{ REG_COM8, COM8_FASTAEC | COM8_AECSTEP | COM8_AGC | COM8_AEC | COM8_AWB },

/* Matrix coefficients */
{ 0x4f, 0x80 }, { 0x50, 0x80 },
{ 0x51, 0 }, { 0x52, 0x22 },
{ 0x53, 0x5e }, { 0x54, 0x80 },
{ 0x58, 0x9e },

{ REG_COM16, COM16_AWBGAIN }, { REG_EDGE, 0 },
{ 0x75, 0x05 }, { REG_REG76, 0xe1 },
{ 0x4c, 0 }, { 0x77, 0x01 },
{ REG_COM13, /*0xc3*/0x48 }, { 0x4b, 0x09 },
{ 0xc9, 0x60 }, /*{REG_COM16, 0x38}*/
{ 0x56, 0x40 },

{ 0x34, 0x11 }, { REG_COM11, COM11_EXP | COM11_HZAUTO },
{ 0xa4, 0x82/*Was 0x88*/ }, { 0x96, 0 },
{ 0x97, 0x30 }, { 0x98, 0x20 },
{ 0x99, 0x30 }, { 0x9a, 0x84 },
{ 0x9b, 0x29 }, { 0x9c, 0x03 },
{ 0x9d, 0x4c }, { 0x9e, 0x3f },
{ 0x78, 0x04 },

/* Extra-weird stuff. Some sort of multiplexor register */
{ 0x79, 0x01 }, { 0xc8, 0xf0 },
{ 0x79, 0x0f }, { 0xc8, 0x00 },
{ 0x79, 0x10 }, { 0xc8, 0x7e },
{ 0x79, 0x0a }, { 0xc8, 0x80 },

```

```

{ 0x79, 0x0b }, { 0xc8, 0x01 },
{ 0x79, 0x0c }, { 0xc8, 0x0f },
{ 0x79, 0x0d }, { 0xc8, 0x20 },
{ 0x79, 0x09 }, { 0xc8, 0x80 },
{ 0x79, 0x02 }, { 0xc8, 0xc0 },
{ 0x79, 0x03 }, { 0xc8, 0x40 },
{ 0x79, 0x05 }, { 0xc8, 0x30 },
{ 0x79, 0x26 },
{ 0xff, 0xff }, /* END MARKER */
};

void initCamera(int reg) {
    arduinoUnolnut();
    camInit();
    setRes();
    setColor();
    wrReg(0x11, reg);
}

void error_led(void) {
    DDRB |= 32; //make sure led is output
    while (1) { //wait for reset
        PORTB ^= 32; // toggle led
        _delay_ms(100);
    }
}

void twiStart(void) {
    TWCR = _BV(TWINT) | _BV(TWSTA) | _BV(TWEN); //send start
    while (!(TWCR & (1 << TWINT))); //wait for start to be transmitted
    if ((TWSR & 0xF8) != TW_START)
        error_led();
}

void twiWriteByte(uint8_t DATA, uint8_t type) {
    TWDR = DATA;
}

```

```

TWCR = _BV(TWINT) | _BV(TWEN);
while (!(TWCR & (1 << TWINT))) {}
if ((TWSR & 0xF8) != type)
    error_led();
}

void twiAddr(uint8_t addr, uint8_t typeTWI) {
    TWDR = addr;//send address
    TWCR = _BV(TWINT) | _BV(TWEN); /* clear interrupt to start transmission */
    while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
    if ((TWSR & 0xF8) != typeTWI)
        error_led();
}

void wrReg(uint8_t reg, uint8_t dat) {
    //send start condition
    twiStart();
    twiAddr(camAddr_WR, TW_MT_SLA_ACK);
    twiWriteByte(reg, TW_MT_DATA_ACK);
    twiWriteByte(dat, TW_MT_DATA_ACK);
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO);//send stop
    _delay_ms(1);
}

static uint8_t twiRd(uint8_t nack) {
    if (nack) {
        TWCR = _BV(TWINT) | _BV(TWEN);
        while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
        if ((TWSR & 0xF8) != TW_MR_DATA_NACK)
            error_led();
        return TWDR;
    }
    else {
        TWCR = _BV(TWINT) | _BV(TWEN) | _BV(TWEA);
        while ((TWCR & _BV(TWINT)) == 0); /* wait for transmission */
        if ((TWSR & 0xF8) != TW_MR_DATA_ACK)

```

```

    error_led();
    return TWDR;
}
}

uint8_t rdReg(uint8_t reg) {
    uint8_t dat;
    twiStart();
    twiAddr(camAddr_WR, TW_MT_SLA_ACK);
    twiWriteByte(reg, TW_MT_DATA_ACK);
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO); //send stop
    _delay_ms(1);
    twiStart();
    twiAddr(camAddr_RD, TW_MR_SLA_ACK);
    dat = twiRd(1);
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO); //send stop
    _delay_ms(1);
    return dat;
}

void wrSensorRegs8_8(const struct regval_list reglist[]) {
    uint8_t reg_addr, reg_val;
    const struct regval_list *next = reglist;
    while ((reg_addr != 0xff) | (reg_val != 0xff)) {
        reg_addr = pgm_read_byte(&next->reg_num);
        reg_val = pgm_read_byte(&next->value);
        wrReg(reg_addr, reg_val);
        next++;
    }
}

void setColor(void) {
    wrSensorRegs8_8(yuv422_ov7670);
}

void setRes(void) {

```

```

wrReg(REG_COM3, 4); // REG_COM3 enable scaling
wrSensorRegs8_8(qqvga_ov7670);
}

void camInit(void) {
    wrReg(0x12, 0x80);
    _delay_ms(100);
    wrSensorRegs8_8(ov7670_default_regs);
    wrReg(REG_COM10, 32); //PCLK does not toggle on HBLANK.
}

void arduinoUnoInut(void) {
    cli(); // disable interrupts

    /* Setup the 8mhz PWM clock
     * This will be on pin 11*/
    DDRB |= (1 << 3); //pin 11
    ASSR &= ~(_BV(EXCLK) | _BV(AS2));
    TCCR2A = (1 << COM2A0) | (1 << WGM21) | (1 << WGM20);
    TCCR2B = (1 << WGM22) | (1 << CS20);
    OCR2A = 0; // (F_CPU)/(2*(X+1))
    DDRC &= ~15; // low d0-d3 camera
    DDRD &= ~252; // d7-d4 and interrupt pins
    _delay_ms(1000);

    // set up twi for 100khz
    TWSR &= ~3; // disable prescaler for TWI
    TWBR = 72; // set to 100khz

    // enable serial
    UBRR0H = 0;
    UBRR0L = 1; // 0 = 2M baud rate. 1 = 1M baud. 3 = 0.5M. 7 = 250k 207 is 9600 baud rate.
    UCSR0A |= 2; // double speed aysnc
    UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Enable receiver and transmitter
    UCSR0C = 6; // async 1 stop bit 8bit char no parity bits
}

```

```

void StringPgm(const char * str) {
    do {
        while (!(UCSR0A & (1 << UDRE0))); //wait for byte to transmit
        UDR0 = pgm_read_byte_near(str);
        while (!(UCSR0A & (1 << UDRE0))); //wait for byte to transmit
    } while (pgm_read_byte_near(++str));
}

static void captureImg(uint16_t wg, uint16_t hg) {
    uint16_t y, x;
    uint8_t buf[320];

    while (!(PIND & 8)); //wait for high -> vsync
    StringPgm(PSTR("*RDY*"));
    while ((PIND & 8)); //wait for low -> vsync

    y = hg;
    while (y--) {
        x = wg;
        uint8_t*b = buf, *b2 = buf;

        while (!(PINB & 1)); //wait for high -> href

        while (x--) {
            while ((PIND & 4)) { //wait for low -> pclk
                if (!(UCSR0A & (1 << UDRE0)) && b2 < b) {
                    UDR0 = *b2++;
                }
            }
        }
        *b++ = (PINC & 15) | (PIND & 240); // get only first byte
        while (!(PIND & 4)) { //wait for high -> pclk
            if (!(UCSR0A & (1 << UDRE0)) && b2 < b) {
                UDR0 = *b2++;
            }
        }
    }
}

```

```

        }
    }

    while ((PIND & 4)) { //wait for low -> pclk
        if (!(UCSR0A & (1 << UDRE0)) && b2 < b) {
            UDR0 = *b2++;
        }
    }

    while (!((PIND & 4))) { //wait for high -> pclk
        if (!(UCSR0A & (1 << UDRE0)) && b2 < b) {
            UDR0 = *b2++;
        }
    }

}

while ((PINB & 1)); //wait for low -> href

while (b2 < b) {
    UDR0 = *b2++;
    while (!(UCSR0A & (1 << UDRE0))); //wait for byte to transmit
}
}

void capture(int width_, int height_){
    char a = UDR0;
    a++;
    captureImg(width_, height_);
}

```

Terminal_pc1.py

```
import os
import time
import serial
import time
import threading
import json
from termcolor import colored
from flask import Flask, Response, redirect, request, url_for, jsonify, send_file,
render_template
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

serial_Arduino = None
stop_thread = False
config = None
string_receiver_arduino = []
send_msg_Arduono = ""

def serial_run():
    global stop_thread
    global string_receiver_arduino
    global send_msg_Arduono
    global serial_Arduino
    if not serial_Arduino.isOpen():
        serial_Arduino.open()
    try:
        serialString = ""
        while True:
            #print("In thread!")
            while not serial_Arduino.isOpen():
                #print("Serial 1 offline")
            if stop_thread:
```

```

        print("stop threaded")
        time.sleep(0.1)
#print("Test")
if serial_Arduino.isOpen():
    try:
        #print("Thread ready!")
        if send_msg_Arduono != "":
            #print("Writed", send_msg_Arduono2)
            serial_Arduino.write(
                str.encode('{0}\n'.format(send_msg_Arduono)))
        print(
            colored(
                (send_msg_Arduono, "Writed: ",
                 str.encode('{0}\n'.format(send_msg_Arduono))),
                "cyan"))
        send_msg_Arduono = ""
    if serial_Arduino.inWaiting():
        #print("Have data!")
        serialString = serial_Arduino.readline()
        print(
            colored(("Data in serial", serialString),
                   "magenta"))
        serial_Arduino.flush()
        #print(serialString.decode().replace("\n", ""))
        if serialString[0] != '\r\n':
            #print("DEF")
            temp = serialString.decode('Ascii').replace(
                '\r\n', "")
            #print(temp, len(temp))
            #print("zzzz")
            try:
                decode_dec_to_data(temp[21:21 + 22])
            except:
                pass
            try:
                decode_three_degree(temp[21:25])

```

```

        except:
            pass
            #print("eiei")
            string_recever_arduino.append(temp)
            #print(string_recever_arduino)

        except:
            time.sleep(0.1)

    if stop_thread:
        print("stop threaded")
        break

except:
    pass
finally:
    serial_Arduino.close()
    if not serial_Arduino.isOpen():
        print("Program,Serial comm is closed")

def loadConfig():
    global config
    with open("config.json") as json_data_file:
        config = json.load(json_data_file)

def init():
    global serial_Arduino
    serial_Arduino = serial.Serial(port=config['port_Arduino1'],
                                    baudrate=config['buadrate_Arduino1'],
                                    parity=serial.PARITY_NONE,
                                    stopbits=serial.STOPBITS_ONE,
                                    bytesize=serial.EIGHTBITS)

def decode_three_degree(code):
    if len(code) == 4 and code[0] == 'X':

```

```

tmp = ("*** Answer from PC2 ***") + '\n'
tmp += ' -45 Degree: ' + str(bin(int(code[1],
                                         16)))[2:].zfill(4) + '\n'
tmp += ' 0 Degree: ' + str(bin(int(code[2],
                                         16)))[2:].zfill(4) + '\n'
tmp += ' 45 Degree: ' + str(bin(int(code[3],
                                         16)))[2:].zfill(4) + '\n'
print(colored(tmp + "*** ----- ***", "green"))

```

```

def decode_dec_to_data(code):
    #print("Call function")
    position = [[40, 20], [40, 70], [100, 20], [100, 70]]
    tmp = ""
    tmp += ("*** Answer from PC2 ***") + '\n'
    if code[0] == '*' and code[-1] == '*' and len(code) == 22:
        #print("Yes this code")
        idx = 0
        idy = 0
        for x in code[1:-1]:
            if idx < 4:
                tmp += (' ') + (
                    '(' + str(position[idy][0] + (idx * 4)) + ',' +
                    str(position[idy][1] +
                         (idx * 4)) + '): ' + str(ord(x))) + '\n'
            else:
                tmp += (
                    'mean of QUADRANT {}: '.format(idy) + str(ord(x))) + '\n'
            idx += 1
            if idx == 5:
                idx = 0
                idy += 1
    print(colored(tmp[:-1] + '\n' + "*** ----- ***", "green"))

```

Server route

```

@app.route('/senddata', methods=['POST'])
def senddata():
    global send_msg_Arduono
    try:
        command = request.get_json()['data']
        #print(command)
        send_msg_Arduono = command
        return 'OK'
    except:
        return 'Error'

#arduino 1
@app.route('/stream1')
def stream():
    def read_process():
        while True:
            global string_receiver_arduino
            for idx in range(len(string_receiver_arduino)):
                if len(string_receiver_arduino):
                    yield "data:" + string_receiver_arduino.pop(0) + '\n\n'
                    time.sleep(0.1)

    return Response(read_process(), mimetype='text/event-stream')

#thread for terminal input
def user_inp():
    global stop_thread
    global send_msg_Arduono
    #try:
    while True:
        tmp = input()
        if len(tmp) == 4:
            try:
                send_msg_Arduono = hex(int(tmp, 2))[2]

```

```

#print(send_msg_Arduono)
except:
    send_msg_Arduono = tmp
else:
    send_msg_Arduono = tmp
#print('command', x)
if send_msg_Arduono == '#' or stop_thread:
    stop_thread = True
    break
#except:
#    stop_thread = True

if __name__ == "__main__":
    loadConfig()
    init()
    thread_serial = threading.Thread(target=serial_run).start()
    thread_inp = threading.Thread(target=user_inp).start()
    send_msg_Arduono = ""
    try:
        app.run(host="0.0.0.0",
                port=5000,
                debug=True,
                use_debugger=False,
                use_reloader=False)
    except:
        pass
    finally:
        stop_thread = True
        if serial_Arduino.isOpen():
            serial_Arduino.close()

```

```

server.py
import json
from flask import Flask, Response, redirect, request, url_for, jsonify, send_file,
render_template
import itertools
import os
import time
import serial
import time
import threading
import signal
from flask_cors import CORS
from Image_processing import *
from read_mode import get_mode
from show_led import led_freestyle, setup_pin
from termcolor import colored

#define zone
PROCESS_WITH_ML = 0
PROCESS_WITH_IFELSE = 1

app = Flask(__name__)
CORS(app)

state = 0
stop_threads = False
send_msg = ""
config = {}

serial_Arduino3 = None # Camera
serial_Arduino2 = None # Servo and FM

string_receiver_arduino2 = []

send_msg_Arduino3 = ""
send_msg_Arduino2 = ""

```

```
processing_mode = 0 # 0: ML , 1: if else

pattern_led = [0, 0, 0]

travel_capture_thread = None

storage_image_id = [-1, -1, -1]

focus_idx = -1
```

```
def serial_arudino2():
    global stop_threads
    global send_msg
    global string_recever_arduino2
    global send_msg_Arduono2
    global serial_Arduino2
    global travel_capture_thread
    global storage_image_id
    global focus_idx
    if not serial_Arduino2.isOpen():
        serial_Arduino2.open()
    try:
        serialString = ""
        while True:
            #print("In thread!")
            while not serial_Arduino2.isOpen():
                #print("Serial 2 offline")
                if stop_threads:
                    break
                time.sleep(0.1)
            #print("Test")
        if serial_Arduino2.isOpen():
            try:
                #print("Thread ready!")

```

```

if send_msg_Arduono2 != "":
    #print("Writed", send_msg_Arduono2)
    serial_Arduono2.write(
        str.encode('{0}\n'.format(send_msg_Arduono2)))
    print(send_msg_Arduono2, "Writed: ",
          str.encode('{0}\n'.format(send_msg_Arduono2)))
    send_msg_Arduono2 = ""
if serial_Arduono2.inWaiting():
    #print("Have data!")
    serialString = serial_Arduono2.readline()
    print(
        colored(("Data in serial", serialString),
               "magenta"))
    serial_Arduono2.flush()
if serialString[0] != '\r\n':
    temp = serialString.decode('Ascii').replace(
        '\n', "")
    print(colored(str(storage_image_id), "red"))
    if "capture" in temp:
        #print("Start capture")
        if not travel_capture_thread.isAlive():
            storage_image_id = [-1, -1, -1]
            travel_capture_thread = threading.Thread(
                target=travel_capture)
            travel_capture_thread.start()
    elif str(storage_image_id) != '[-1, -1, -1]':
        #print("Do this")
        #print(temp)
        try:
            if (temp[21] >= '0' and temp[21] <= '9'
                ) or (temp[21] >= 'a'
                      and temp[21] <= 'f'):
                #print(" Fucking Doing")
                focus_idx = storage_image_id.index(
                    int(temp[21], 16))
                if not travel_capture_thread.isAlive():

```

```

travel_capture_thread = threading.Thread(
    target=travel_capture)
travel_capture_thread.start()
except:

    #print("Err")
    pass
    string_receiver_arduino2.append(temp)
except:
    time.sleep(0.1)
if stop_threads:
    break

except:
    pass
finally:
    print("Something wrong!")
    serial_Arduino2.close()
    if not serial_Arduino2.isOpen():
        print("Program,Serial comm is closed")

```

```

@app.route('/senddata', methods=['POST'])
def senddata():
    global send_msg_Arduono2
    global storage_image_id
    global travel_capture_thread
    command = request.get_json()['data']
    #print(command)
    if command == 'TEST':
        if not travel_capture_thread.isAlive():
            storage_image_id = [-1, -1, -1]
            travel_capture_thread = threading.Thread(target=travel_capture)
            travel_capture_thread.start()
            #travel_capture_thread.start()
    else:

```

```

    send_msg_Arduino2 = command
    return 'OK'

#arduino 2
@app.route('/stream2')
def stream():
    def read_process():
        while True:
            global string_receiver_arduino2
            for idx in range(len(string_receiver_arduino2)):
                if len(string_receiver_arduino2):
                    yield "data:" + string_receiver_arduino2.pop(0) + '\n\n'
                    time.sleep(0.1)

    return Response(read_process(), mimetype='text/event-stream')

@app.route('/')
def index():
    return render_template('main.html')

def loadConfig():
    global config
    with open("config.json") as json_data_file:
        config = json.load(json_data_file)
    #print(config)

def init_Serial():
    global serial_Arduino3
    global serial_Arduino2
    global config
    print(config)

```

```

serial_Arduino2 = serial.Serial(
    port=config['port_Arduino2'],
    baudrate=config['buadrate_Arduino2'],
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS)

serial_Arduino3 = serial.Serial(
    port=config['port_Arduino3'],
    baudrate=config['buadrate_Arduino3'],
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE,
    bytesize=serial.EIGHTBITS,
)

if serial_Arduino3.isOpen():
    #print("Close port 3")
    serial_Arduino3.close()
    #print('Wait for Arduino to set itself up...')
    time.sleep(5)

def travel_capture():
    global focus_idx
    global send_msg_Arduno2
    global serial_Arduino3
    global serial_Arduino2
    global processing_mode
    global pattern_led
    global storage_image_id
    processing_mode = get_mode()
    DEGREE = ["L", "M", "R"]
    ans = []
    loadding_led()
    if focus_idx == -1:
        for j in range(3):

```

```

if serial_Arduino3.isOpen():
    serial_Arduino3.close()
    #print("Close port 3")
if not serial_Arduino2.isOpen():
    serial_Arduino2.open()
    #print("Open port 3")
send_msg_Arduono2 = DEGREE[j]
#print("before delay")
time.sleep(2)
#print("After delay")
if serial_Arduino2.isOpen():
    #print("Close port 2")
    serial_Arduino2.close()
if not serial_Arduino3.isOpen():
    serial_Arduino3.open()
    #print("Open port 3")
if processing_mode == PROCESS_WITH_ML:
    print(colored("processing with ML", "yellow"))
    code, pattern = process_with_ml(serial_Arduino3, DEGREE[j],
                                    config['server_processing'])
elif processing_mode == PROCESS_WITH_IFELSE:
    print(colored('processing with if else', "yellow"))
    code, pattern = process_with_ifelse(serial_Arduino3, DEGREE[j])
ans.append(code)
print(colored(code, "magenta"))
pattern_led[2 - j] = int(code)
#print(colored(pattern_led, "blue"))
if serial_Arduino3.isOpen():
    serial_Arduino3.close()
if not serial_Arduino2.isOpen():
    serial_Arduino2.open()
storage_image_id = ans
print(colored(("Storage image id: ", storage_image_id), "cyan"))
send_msg_Arduono2 = 'X' + ''.join([hex(x)[2:] for x in ans])
print(colored("Send", "green"), colored(send_msg_Arduono2, "cyan"),
      colored("To arduino1", "green"))

```

```

#print(ans)
else: ## forcus on index
    if serial_Arduino3.isOpen():
        serial_Arduino3.close()
        #print("Close port 3")
    if not serial_Arduino2.isOpen():
        serial_Arduino2.open()
        #print("Open port 3")
    send_msg_Arduono2 = DEGREE[focus_idx]
    #print("before delay")
    time.sleep(2)
    #print("After delay")
    if serial_Arduino2.isOpen():
        #print("Close port 2")
        serial_Arduino2.close()
    if not serial_Arduino3.isOpen():
        serial_Arduino3.open()
        #print("Open port 3")
    if processing_mode == PROCESS_WITH_ML:
        print(colored("processing with ML", "yellow"))
        code, pattern = process_with_ml(serial_Arduino3, DEGREE[focus_idx],
                                         config['server_processing'])
    elif processing_mode == PROCESS_WITH_IFELSE:
        print(colored('processing with if else', "yellow"))
        code, pattern = process_with_ifelse(serial_Arduino3,
                                             DEGREE[focus_idx])
    ans.append(code)
    print(colored(pattern, "magenta"))
    pattern_led[2 - focus_idx] = int(code)
    #print(colored(pattern_led, "blue"))
    if serial_Arduino3.isOpen():
        serial_Arduino3.close()
    if not serial_Arduino2.isOpen():
        serial_Arduino2.open()
    send_msg_Arduono2 = encode_dec_to_data(pattern)
    print(colored("Send", "green"), colored(send_msg_Arduono2, "cyan"),

```

```
    colored("To arduino1", "green"))
time.sleep(1)
focus_idx = -1
send_msg_Arduono2 = 'C'
```

```
def encode_dec_to_data(arr):
    tmp = ""
    arr = arr[0] + arr[1] + arr[2] + arr[3]
    #print(arr)
    for x in arr:
        #print(x, str(chr(x)))
        tmp += str(chr(x))
    #print(str.encode(tmp))
    return '*' + tmp + '*'
```

```
def init_Camera():
    global serial_Arduino3
    # Flush 5 image
    for j in range(4):
        print("Flush ", j + 1)
        flush_image(serial_Arduino3)
```

```
def serial_arudino3():
    #init_Camera()
    pass
```

```
def loadding_led():
    global pattern_led
    pattern_led[0] = 100
    pattern_led[1] = 100
    pattern_led[2] = 100
```

```

if __name__ == "__main__":
    setup_pin()
    if processing_mode == PROCESS_WITH_IFELSE:
        print(colored("PROCESS WITH IF ELSE", "yellow"))
    else:
        print(colored("PROCESS WITH ML", "yellow"))
    loadConfig()
    init_Serial()
    travel_capture_thread = threading.Thread(target=travel_capture)
    thread_Serial2 = threading.Thread(target=serial_arudino2)
    loadding_led()
    thread_for_show_led = threading.Thread(target=led_freestyle,
                                            args=(pattern_led, stop_threads))
    thread_Serial2.start()
    thread_for_show_led.start()
    try:
        app.run(host="0.0.0.0",
                port=5000,
                debug=True,
                use_debugger=False,
                use_reloader=False)
    except:
        print("Error!")
    finally:
        stop_threads = True
        if serial_Arduino2.isOpen():
            serial_Arduino2.close()
            print("User,Serial comm2 is closed")
        if serial_Arduino3.isOpen():
            serial_Arduino3.close()
            print("User,Serial comm3 is closed")

```

send_to_server.py

```
from _future_ import print_function
import requests
import json
import cv2

def send(target_ip, image_file):
    addr = 'http://{}:5500'.format(target_ip)
    test_url = addr + '/api/test'
    content_type = 'image/jpeg'
    headers = {'content-type': content_type}
    img = cv2.imread(r'{}'.format(image_file))
    _, img_encoded = cv2.imencode('.jpg', img)
    response = requests.post(test_url,
                             data=img_encoded.tostring(),
                             headers=headers)
    return json.loads(response.text)
```

```

server_image.py
import cv2
from PIL import Image
import sys
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import jsonpickle
import time
import json
from flask_cors import CORS
from flask import Flask, Response, redirect, request, url_for, jsonify, send_file,
render_template
app = Flask(__name__)
CORS(app)

model = load_model('./ML/model_datacom')

def process(image_):
    array_test_img = image.img_to_array(image_)
    array_test_img = np.expand_dims(array_test_img, axis=0)
    target = np.vstack([array_test_img])
    return np.argmax(model.predict(target), axis=-1)

def process_image(img):
    return process(img)[0]

def process_with_ml(img):
    ans1 = "x"
    ans2 = "y"
    while ans1 != ans2:
        ans1 = process_image(img)
        time.sleep(0.5)

```

```
ans2 = process_image(img)
return ans1

@app.route('/api/test', methods=['POST'])
def test():
    r = request
    nparr = np.fromstring(r.data, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    return str(process_with_ml(img))

if __name__ == "__main__":
    app.run(host="0.0.0.0",
            port=5500,
            debug=True,
            use_debugger=False,
            use_reloader=False)
```

image_processing.py

```
import cv2
from
    import Image
import sys
import numpy as np
import time
from send_to_server import *

width = 160
height = 120

def process_without_ML(image):
    img = cv2.imread(r'{}'.format(image), 0)
    #print("loaded image")
    ans = []
    tmp = []
    for i in range(4):
        tmp.append(
            int(img[40 + (i * 4):40 + (i * 4) + 1,
                   20 + (i * 4):20 + (i * 4) + 1].mean()))
    ans.append(tmp)
    tmp = []
    for i in range(4):
        tmp.append(
            int(img[40 + (i * 4):40 + (i * 4) + 1,
                   70 + (i * 4):70 + (i * 4) + 1].mean()))
    ans.append(tmp)
    tmp = []
    for i in range(4):
        tmp.append(
            int(img[100 + (i * 4):100 + (i * 4) + 1,
                   20 + (i * 4):20 + (i * 4) + 1].mean()))
    ans.append(tmp)
```

```

tmp = []
for i in range(4):
    tmp.append(
        int(img[100 + (i * 4):100 + (i * 4) + 1,
              70 + (i * 4):70 + (i * 4) + 1].mean()))
ans.append(tmp)
pattern = ""
for x in ans:
    avg = int(sum(x) / len(x))
    x.append(avg)
    pattern += '0' if avg < 70 else '1'
#print(ans, pattern)
return ans, pattern

```

```

def process_with_ml(serial, x, target_ip):
    ans1 = "x"
    ans2 = "y"
    while ans1 != ans2:
        ans1 = process_image(serial, x, target_ip)
        time.sleep(0.5)
        ans2 = process_image(serial, x, target_ip)
        print(ans1, ans2)
    avg, ansX = process_without_ML('./{}.bmp'.format(x))
    return ans1, avg

```

```

def process_image(serial, x, target_ip):
    read_image(serial, x)
    read_image(serial, x)
    read_image(serial, x)
    read_image(serial, x)
    read_image(serial, x)
    return send(target_ip, './{}.bmp'.format(x))

```

```

def process_with_ifelse(serial, x):
    ans1 = "x"
    ans2 = "y"
    avg = []
    while ans1 != ans2:
        read_image(serial, x)
        avg, ans1 = process_without_ML('./{}.bmp'.format(x))
        time.sleep(0.5)
        avg, ans2 = process_without_ML('./{}.bmp'.format(x))
        print(ans1, ans2)
    return ans1, avg

```

```

def read_image(serial, x):
    #print("wait for arduino rdy")
    serial.flush()
    serial.read_until('*RDY*'.encode('utf-8'))
    #print("captured")
    serial.write(str.encode('X'))
    data = serial.read(width * height)
    _image = Image.frombytes('P', (width, height), data)
    _image = _image.transpose(Image.ROTATE_270)
    _image.save('./{}.bmp'.format(x))
    #print('save image')

def flush_image(serial):
    #print("wait for arduino rdy")
    #time.sleep(0.1)
    serial.flush()
    serial.read_until('*RDY*'.encode('utf-8'))
    print("captured")
    serial.write(str.encode('X'))
    print("flush")
    print("Reading Image...")
    data = serial.read(width * height)

```

read_mode.py

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
pin_select_mode = 23
GPIO.setup(pin_select_mode, GPIO.IN)

def get_mode():
    return not GPIO.input(pin_select_mode)
```

led_rpi.py

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
import random
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BCM) # Use physical pin numbering

led_lst = [26, 19, 13, 6, 5, 11, 9, 10, 22, 27, 17, 18]

pin_select_mode = 23

for x in led_lst:
    GPIO.setup(x, GPIO.OUT, initial=GPIO.HIGH)
GPIO.setup(pin_select_mode, GPIO.IN)

def freestyle(pattern):
    for k in range(12):
        GPIO.output(led_lst[k], GPIO.LOW)
        sleep(0.01)
        GPIO.output(led_lst[k], GPIO.HIGH)
        sleep(0.01)
    for j in range(3):
        #print(x,end=' ')
        for i in range(4):
            #print(i, (4*(j+1))-i-1)
            GPIO.output(led_lst[(4 * (j + 1)) - i - 1],
                        GPIO.HIGH if pattern[j] & 1 else GPIO.LOW)
            pattern[j] >>= 1
    #print()

def clear():
    for k in range(11, -1, -1):
        GPIO.output(led_lst[k], GPIO.LOW)
        sleep(0.01)
```

```
GPIO.output(led_lst[k], GPIO.HIGH)
sleep(0.01)

while True:
    #print(type(GPIO.input(pin_select_mode)))
    t = 15 if GPIO.input(pin_select_mode) == 0 else 0
    #print(t)
    freestyle([t, t, t])
    sleep(2)
    clear()
```

show_led.py

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
import random
import threading
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BCM) # Use physical pin numbering

led_lst = [26, 19, 13, 6, 5, 11, 9, 10, 22, 27, 17, 18]

virtual_led = [0 for _ in range(12)]

def setup_pin():
    #print('setup pin')
    for x in led_lst:
        GPIO.setup(x, GPIO.OUT, initial=GPIO.HIGH)
    # pass

def onLED(n):
    GPIO.output(led_lst[n], GPIO.LOW)
    # print('LED :', n, "ON")
    # virtual_led[n] = 1

def offLED(n):
    GPIO.output(led_lst[n], GPIO.HIGH)
    # print('LED :', n, "OFF")
    # virtual_led[n] = 0

# 0-15 : display number
# other : loading
```

```

def led_freestyle(pattern, stop_threads):

    run, direction = [4,1,8], [0,0,0]
    #print("In thread")
    while not stop_threads:
        for j in range(3):
            #print(x,end=' ')
            if pattern[j] < 0 or pattern[j] > 15: # running LED
                x = run[j]
                for i in range(4):
                    if x & 1:
                        onLED(4 * j + i)
                    else:
                        offLED(4 * j + i)
                    x >>= 1
                    #print(x)
                if run[j] == 8:
                    direction[j] = 1
                elif run[j] == 1:
                    direction[j] = 0
                if direction[j] == 0:
                    run[j] <<= 1
                else:
                    run[j] >>= 1
                #print()
            else:
                x = pattern[j]
                for i in range(4):
                    if x & 1:
                        onLED(4 * j + i)
                    else:
                        offLED(4 * j + i)
                    x >>= 1
                sleep(0.25)
                # print()
                # print(' 0 1 2 3 4 5 6 7 8 9 10 11')

```

```
# print(virtual_led)

def clear():
    for k in range(11, -1, -1):
        onLED(k)
        sleep(0.01)
        offLED(k)
        sleep(0.01)
```