****

**รายงาน**

**เรื่อง การถ่ายภาพและส่งข้อมูลผ่านคลื่น FM**

**กลุ่มที่ 1**

**ผู้จัดทำ**

1. นายธนพล วงศ์อาษา กลุ่มที่ 8 รหัสนักศึกษา 62010356
2. นายสุทธิราช ภูโท กลุ่มที่ 8 รหัสนักศึกษา 62010966
3. นายธนดล สินอนันต์วณิช กลุ่มที่ 22 รหัสนักศึกษา 62010345
4. นายธนา ติ้งประสม กลุ่มที่ 22 รหัสนักศึกษา 62010381
5. นายธีรดนย์ จันทร์หอม กลุ่มที่ 44 รหัสนักศึกษา 62010436
6. นายสิรวิชญ์ สุขวัฒนาวิทย์ กลุ่มที่ 44 รหัสนักศึกษา 62010948

**อาจารย์ที่ปรึกษา**

รศ.ดร. อรฉัตร จิตต์โสภักตร์

ดร. จิระศักดิ์ สิทธิกร

รายงานนี้เป็นส่วนหนึ่งในรายวิชา Data Communications (01076007)

ภาคเรียนที่ 1 ปีการศึกษา 2563 ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

# **คำนำ**

รายงานนี้เป็นส่วนหนึ่งในรายวิชา Data Communications (01076007) โดยมีจุดประสงค์ เพื่อให้ได้ศึกษาความรู้ในเรื่องการรับส่งข้อมูลผ่านคลื่น FM ได้อย่างเข้าใจและมีการปฏิบัติจริง โดยรายงานฉบับนี้มีเนื้อหาเกี่ยวกับการออกแบบการทำงานของระบบควบคุมกล้อง โดยมีการส่งข้อมูลผ่านสัญญานคลื่น FM โดยในงานชิ้นนี้มีการใช้ความรู้ในเรื่องของการส่งข้อมูลแบบ Stop and Wait ARQ, การแปลงคลื่นสัญญาน (Digital Modulation) แบบ FSK (Frequency Shift Keying), การออกแบบ Frame ในการส่งข้อมูล (Flow Control) และการควบคุมความผิดพลาดของข้อมูล (Error Control)

คณะผู้จัดทำหวังว่ารายงานเล่มนี้จะเป็นประโยชน์แก่ผู้อ่าน นักเรียนหรือนักศึกษาที่กำลังศึกษาเรื่องนี้อยู่ หากมีข้อเสนอแนะหรือข้อผิดพลาดประการใด ทางผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้

คณะผู้จัดทำ

# 

# **สารบัญ**

**เรื่อง หน้า**

[**คำนำ**](#_xerysr3sxru0) **1**

[**สารบัญ**](#_ue1j614rnwmf) **2**

[**บทที่ 1  
บทนำ**](#_qohecfx7yhqz) **1**

[ที่มาและความสำคัญ](#_retrr0eptnmv) 1

[วัตถุประสงค์](#_y5o22d2l0hm1) 1

[ขอบเขตของโครงงาน](#_5a4azijg4uey) 1

[**บทที่ 2  
เอกสารที่เกี่ยวข้อง**](#_ubmtdhu006cd) **4**

[การเข้ารหัสสัญญาน (Signal Modulation)](#_p6l51foezdo2) 4

[การมอดูเลตสัญญาณดิจิตอล](#_g88ruusk7oqo) 4

[Bit Rate and Baud Rate](#_yrgmz0gcbos6) 5

[Carrier Signal](#_k6s5tce7balp) 5

[การมอดูเลตทางความถี่ (Frequency-Shift Keying : FSK)](#_5ubjzw9fc1hu) 6

[การควบคุมอัตราการไหลของข้อมูล (Flow Control)](#_ts42ml4vcu94) 7

[การควบคุมความผิดพลาดของข้อมูล (Error Control)](#_bgu9bgam0buy) 7

[กลไกในการควบคุมอัตราการไหลและควบคุมความผิดพลาดของข้อมูล](#_mywce5flfdfz) 8

[Stop-and-Wait ARQ](#_6ulouvjh5obb) 8

[Machine Learning (ML) for detect images](#_tagbcwkh61nf) 9

[Convolutional Neural Network (CNN)](#_qsacvbvlkkrr) 9

[Feature Extraction](#_pfnj51vbt6wd) 10

[Max Pooling](#_3g9ba27ij77c) 10

[การติดตั้งกล้อง OV7670 Camera Module](#_j5xlce55d9kv) 12

[การสร้าง Server ด้วย python flask](#_5e7kq8o1vgcq) 16

[ปัญหาครอสทอล์ค (Crosstalk)](#_5ng52j5rg8bz) 17

[**บทที่ 3  
ขั้นตอนและวิธีการดำเนินงาน**](#_hfa3mwsn49cl) **18**

[ขั้นตอนการดำเนินงาน](#_2ik1ddynsrm7) 18

[ระยะเวลาดำเนินงาน](#_wjz5ne7opw2m) 18

[การออกแบบระบบ](#_5hl6i0cia6x) 18

[วัสดุอุปกรณ์และโปรแกรมที่ใช้ (Hardware & Software)](#_y2inb4fthq7m) 39

[**บทที่ 4  
ผลการดำเนินงาน**](#_aasd2uv3438l) **41**

[ผลลัพท์การประมวลผลภาพด้วย Machine Learning (ML)](#_ldeuzi4euwtu) 50

[**บทที่ 5  
สรุปผลการดำเนินงาน**](#_861pa4jtg32d) **54**

[สรุปผลการดำเนินงาน](#_a9oj11221jul) 54

[ปัญหาอุปสรรคและแนวทางแก้ไข](#_r0wpnm3201fg) 54

[**บทที่ 6**](#_ry200iorz9t2) **56**

[**สิ่งที่ทำได้เพิ่มเติมหลังการทดลอง**](#_tjh48yg7vhb4) **56**

[**บรรณานุกรม**](#_hiufbansi2t2) **57**

[**ภาคผนวก**](#_efzo979li4bt) **59**

# บทที่ 1 บทนำ

## ที่มาและความสำคัญ

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

## วัตถุประสงค์

โครงงานเรื่อง ถ่ายภาพส่งสัญญานผ่านคลื่น FM มีวัตถุประสงค์ในการศึกษาค้นคว้า ดังนี้

1. เพื่อศึกษาการทำงานส่งข้อมูลผ่าน Frequency Modulation (FM)
2. ฝึกการออกแบบ Frame ในชั้น Data Link Layer (OSI Model)
3. ฝึกการออกแบบ Flow control และ Error control ของเฟรมที่ส่ง

## ขอบเขตของโครงงาน



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

1. **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

1. **ส่วน Tx, Rx, Camera**

* ใช้เป็น Arduino

1. **ส่วนการเชื่อมต่อระหว่าง Arduino <-> Raspberry Pi**

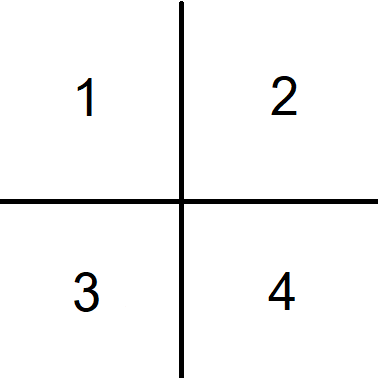
* เป็น สาย USB 2.0 Type B

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

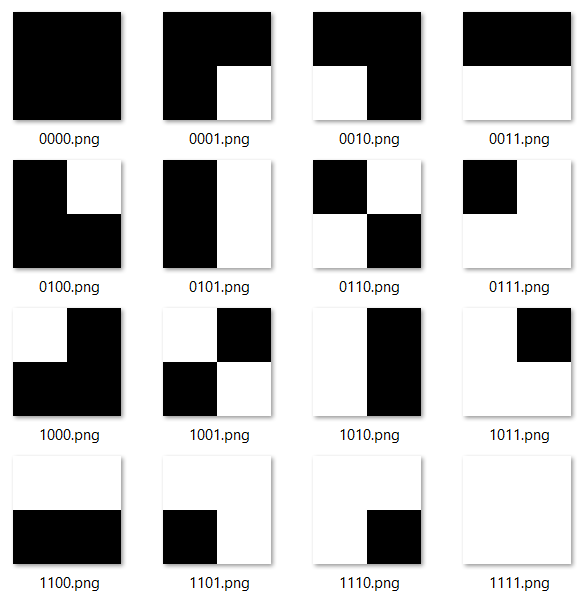
* ต้องใช้ Digital Modulation ที่เหมาะสม
* ส่งแบบ Wireless (FM)
* มี Error Detection
* มีการกำหนด Frame Design ที่เหมาะสม
* มีการกำหนด Flow & Error Control

1. **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

## การเข้ารหัสสัญญาน (Signal Modulation)

การมอดูเลต (Modulation) จะเป็นหลักการที่สำคัญที่ใช้ในการสร้างสัญญาณอนาล็อก ก่อนที่จะถูกส่งออกไป ซึ่งมอดูเลตชันจะแบ่งออกเป็น 2 ประเภทคือ การมอดูเลตสัญญาณดิจิตอล เช่น ASK (amplitude shift keying), FSK (frequency shift keying), PSK (phase shift keying) และการมอดูเลตอนาล็อก เช่น AM (amplitude modulation), FM (frequency modulation), PM (phase modulation) นอกจากนั้นแล้วจะได้พูดถึงเรื่องของโมเด็ม ซึ่งเป็นอุปกรณ์ที่ใช้ในการส่งสัญญาณอนาล็อกผ่านสายโทรศัพท์

## การมอดูเลตสัญญาณดิจิตอล

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

คุณลักษณะของคลื่นรูป Sine จะประกอบไปด้วย แอมพลิจูด, ความถี่ และเฟส ถ้าคุณลักษณะอย่างใดอย่างหนึ่งมีการเปลี่ยนแปลง นั่นหมายความว่าจะได้คลื่นรูป Sine คลื่นใหม่ที่ได้มีการเปลี่ยนแปลงไปจากเดิม จากหลักการนี้จึงสามารถนำมาประยุกต์ใช้กับสัญญาณดิจิตอลได้ เช่น การเปลี่ยนแปลงคุณลักษณะของคลื่นรูป Sine แต่ละครั้งจะหมายถึงบิตข้อมูลที่ได้มีการเปลี่ยนแปลงไป ดังนั้นในการแปลงบิตข้อมูลเป็นสัญญาณอนาล็อกนั้น จะหมายถึงการเปลี่ยนคุณลักษณะของคลื่นรูป Sine นั่นเอง หรือเรียกอีกอย่างว่า การมอดูเลตสัญญาณดิจิตอล 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)

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

### 

### 

### Bit Rate and Baud Rate

ในการมอดูเลตสัญญาณดิจิตอลนั้น มีคำที่ใช้งานกันอยู่บ่อย ๆ คือ อัตราบิต และอัตราบอด

* **อัตราบิต (Bit Rate)** หมายถึง จำนวนของบิตข้อมูลที่สามารถส่งได้ใน 1 วินาที มีหน่วยเป็น bit per second (bps)
* **อัตราบอด (Baud Rate)** หมายถึง อัตราของการเปลี่ยนแปลงสัญญาณใน 1 วินาที ในการเปลี่ยนแปลงสัญญาณในแต่ละครั้งนั้นสามารถส่งข้อมูลหนึ่งบิตหรือมากกว่านั้นก็ได้ เช่น ถ้าใน 1 วินาที สัญญาณมีการเปลี่ยนเฟส 5 ครั้ง อัตราบอดของสัญญาณนี้จะเท่ากับ 5 บอดต่อวินาที (baud/s)

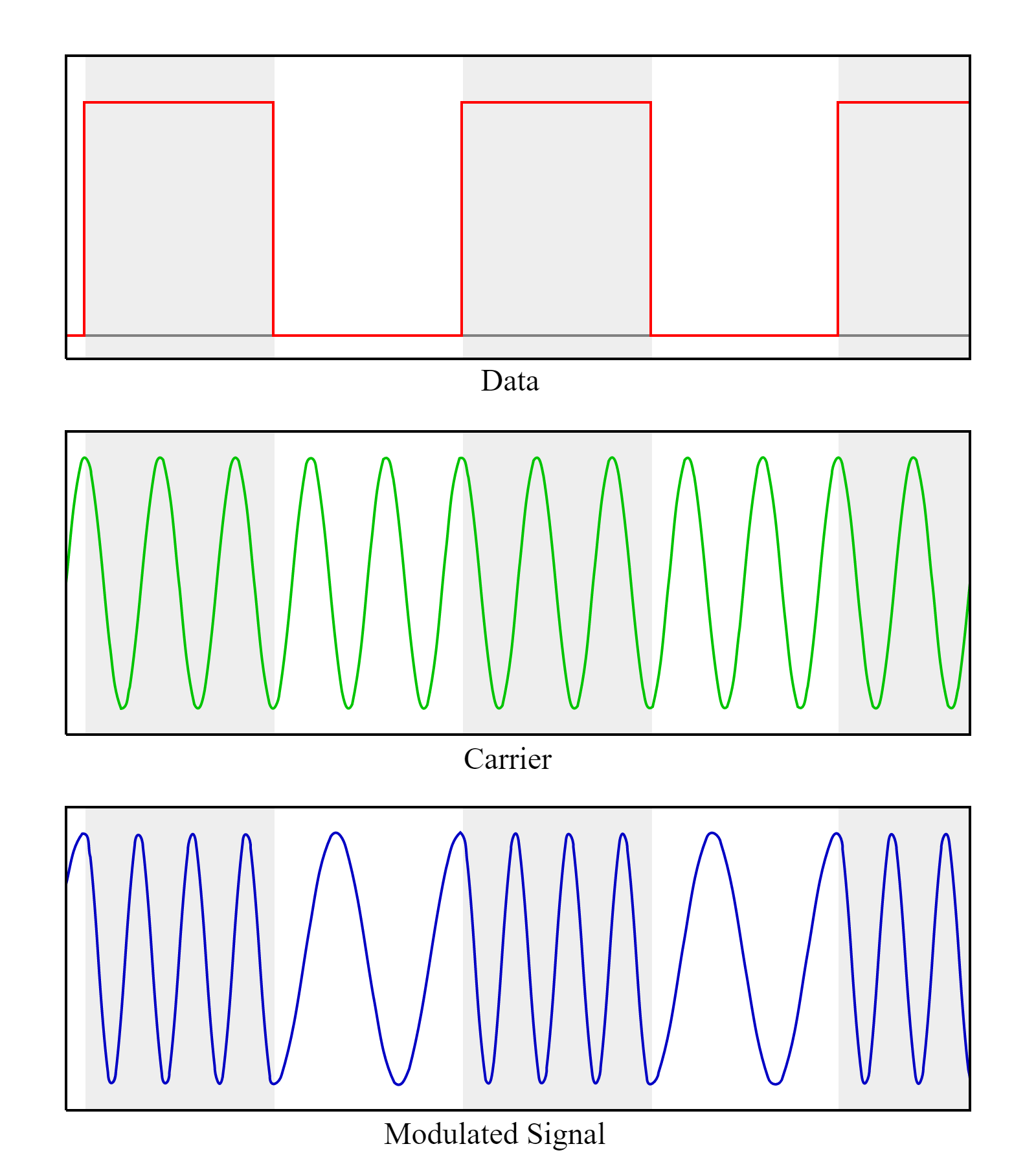
อัตราบอด จะมีค่าเท่ากับหรือน้อยกว่า อัตราบิต ถ้าจะเปรียบเทียบระหว่างอัตราบิต อัตราบอด กับการเดินทางด้วยรถยนต์แล้ว อัตราบอดจะเปรียบเสมือนรถยนต์ ส่วนอัตราบิตจะเปรียบเสมือนผู้โดยสาร ถ้าสมมติให้รถยนต์ 1,000 คัน และะแต่ละคันไม่มีผู้โดยสารคนอื่น นอกจากคนขับ ต้องการเดินทางจากที่หนึ่งไปยังอีกที่หนึ่ง ซึ่งการเดินทางครั้งนี้จะมีคนที่สามารถเดินทางได้ทั้งหมด 1,000 คน แต่ถ้ารถยนต์แต่ละคัน มีผู้โดยสารเพิ่มอีกคันละ 3 คน (ถ้ารวมคนขับด้วยเป็น 4) การเดินทางครั้งนี้จะมีคนที่สามารถเดินทางได้ทั้งหมด 4,000 คน จากตัวอย่างที่ได้กล่าวไปนั้นจะเห็นได้ว่า ปัญหาของการจราจรจะขึ้นอยู่กับจำนวนของรถบนท้องถนน ไม่ได้ขึ้นอยู่กับจำนวนของคนโดยสาร เช่นเดียวกันกับการส่งสัญญาณผ่านสื่อกลาง ซึ่งค่าที่เป็นตัวบ่งบอกถึงแบนด์วิดธ์ที่ต้องการนั้นคืออัตราบอด ไม่ใช่อัตราบิต

### **Carrier Signal**

สัญญาณคลื่นพาห์ (Carrier signal) เป็นสัญญาณที่ถูกสร้างขึ้นมาเพื่อที่จะใช้ในการรวมกับสัญญาณของข้อมูล เนื่องจากสัญญาณของข้อมูลอาจจะมีค่าความถี่ไม่เหมาะสมกับช่องสัญญาณ ดังนั้นอุปกรณ์ที่จะทำการส่งสัญญาณอนาล็อกนั้นจะต้องสร้างสัญญาณคลื่นพาห์ขึ้นมาก่อน ซึ่งจะมีความถี่ที่เหมาะสมกับช่องสัญญาณนั้น ๆ จากนั้นจึงนำสัญญาณคลื่นพาห์นี้รวมกับสัญญาณของข้อมูล ซึ่งจะทำให้สัญญาณคลื่นพาห์มีคุณลักษณะที่เปลี่ยนไปจากเดิม เช่น ค่าแอมพลิจูด ความถี่ หรือเฟส เป็นต้น กระบวนการรวมสัญญาณคลื่นพาห์กับสัญญาณของข้อมูล จะเรียกว่า มอดูเลตชัน (Modulation of shift keying) ส่วนสัญญาณของข้อมูลเราสามารถเรียกอีกอย่างได้ว่า Modulating Signal

## การมอดูเลตทางความถี่ (Frequency-Shift Keying : FSK)

เทคนิคการมอดูเลตข้อมูลบิตข้อมูลกับสัญญาณคลื่นพาห์แบบ FSK นั้นจะทำให้ความถี่เปลี่ยนแปลงไปตามบิตของข้อมูล โดยให้แอมพลิจูด และเฟสคงที่ เช่น ถ้าบิตมีค่า ‘1’ จะให้ความถี่มีค่าสูงกว่าปกติ และบิตมสีค่า ‘0’ จะให้ความถี่มีค่าน้อยกว่าปกติ โดยการส่งสัญญาณแบบ FSK จะสามารถทนต่อสัญญาณรบกวนได้ดีกว่า ASK เนื่องจากอุปกรณ์ที่รับข้อมูลไม่ต้องสนใจว่าแอมพลิจูดหรือแรงดันไฟฟ้าจะมมีค่าที่เปลี่ยนแปลงไปอย่างไร จะพิจารณาเฉพาะความถี่เท่านั้น แต่ FSK จะมีข้อเสียที่สือที่ใช้ในการส่งสัญญาณนั้นจะต้องมีแบนด์วิดธ์ที่เพียงพอที่จะสามารถส่งผ่านความถี่ของสัญญาณที่ต้องการได้



รูปที่ 2.1 แสดงภาพสัญญาณการส่งข้อมูลแบบ FSK

## การควบคุมอัตราการไหลของข้อมูล (Flow Control)

การควบคุมอัตราการไหลของข้อมูล เป็นการกล่าวถึงขั้นตอนกระบวนการที่ควบคุมจำนวนของข้อมูลที่ส่งออกไปให้อยู่ในปริมาณที่เหมาะสม ก่อนที่จะได้รับการตอบรับยืนยันจากผู็รับข้อมูล นั่นหมายความว่าข้อมูลที่ถูกส่งไปนั้นจะต้องถูกจำกัดเอาไว้จำนวนหนึ่ง ถ้ายังไม่ได้รับการตอลรับจากผู้รับว่าได้ข้อมูลชุดนั้น ๆ แล้ว ผู้ส่งจะต้องไม่ส่งข้อมูลชุดต่อไป สาเหตุที่ต้องมีการควบคุมอัตราการไหลก็เนื่องจากว่าผู้รับอาจมีความเร็วในการรับข้อมูลไม่เท่ากับผู้ส่ง หรือมีหน่วยความจำในการเก็บข้อมูลอย่างจำกัด หรือมีความเร็วในการประมวลผลต่ำ ดังนั้นการควบคุมอัตราการไหลของข้อมูลจึงมีความจำเป็นอย่างมาก ถ้าผู้รับยังไม่พร้อมจะรับข้อมูลในขณะนั้น ผู้ส่งจะต้องทำการหยุดส่งข้อมูลชั่วคราวเสียก่อน หรือถ้าผู้รับมึความเร็วในการประมวลผลต่ำ ผู้ส่งจะต้องส่งข้อมูลด้วยอัตราเร็วที่เหมาะสม เพื่อให้เกิดความสมดุลในการรับและส่งข้อมูล

## การควบคุมความผิดพลาดของข้อมูล (Error Control)

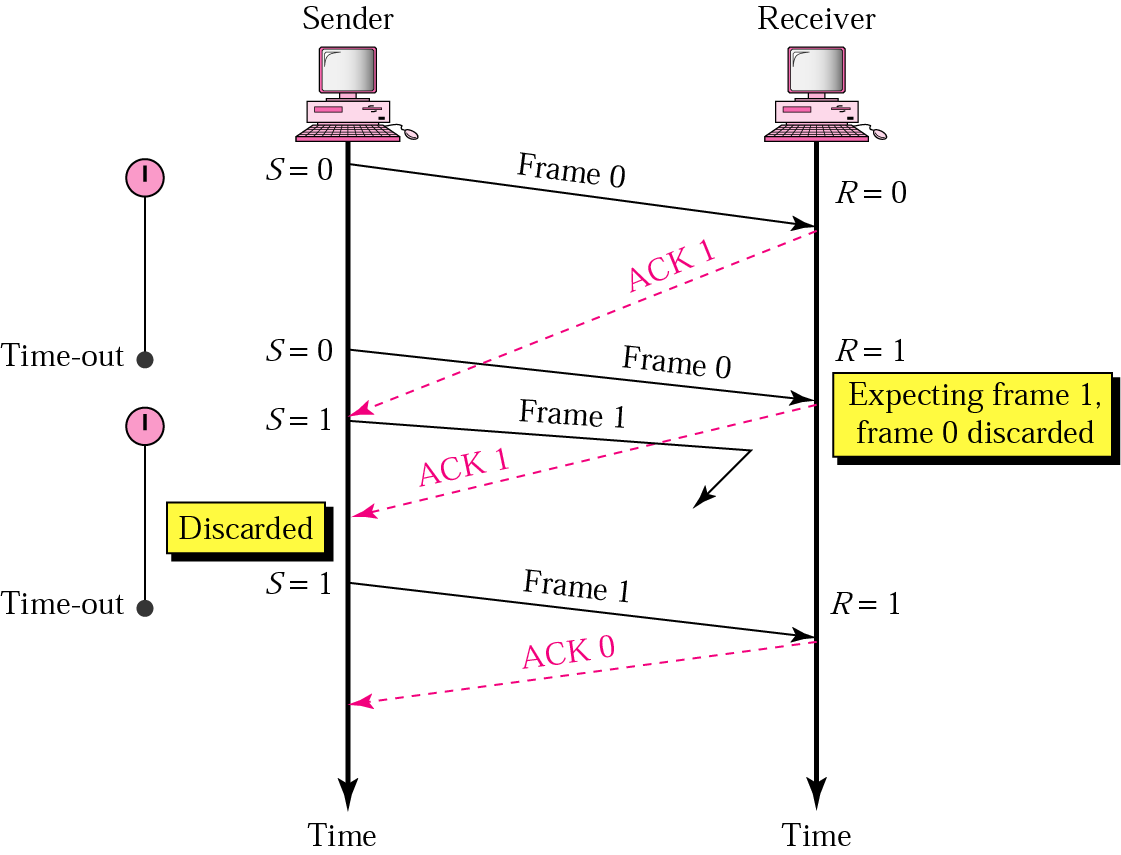
ในเดทาลิงก์เลเยอร์นั้น การควบคุมความผิดพลาดของข้อมูลจะหมายถึง การที่ผู้ส่งต้องส่งข้อมูลไปใหม่อีกครั้งหนึ่ง ถ้าผู้รับไม่สามารถรับข้อมูลหรือได้รับข้อมูลที่ไม่ถูกต้อง สาเหตุที่ต้องมีการควบคุมเนื่องจากว่าข้อมูลจะต้องเดินทางจากที่หนึ่งไปยังอีกที่หนึ่ง จึงมีความเป็นไปได้ที่ข้อมูลชุดนั้นจะเกิดสูญหายหรือเสียหายในระหว่างการเดินทางได้ ดังนั้นผู้รับจะต้องมีกระบวนการในการตรวจสอบความผิดพลาดของข้อมูลนอกจากนั้นจะต้องมีการส่งข้อความไปบอกกับผู้ส่งข้อมูลด้วยว่าข้อมูลชุดไหนที่ได้รับมามีความผิดพลาด ซึ่งผู้ส่งจะต้องทำการส่งข้อมูลชุดนั้น ๆ ไปใหม่อีกครั้งหนึ่ง

## กลไกในการควบคุมอัตราการไหลและควบคุมความผิดพลาดของข้อมูล

### Stop-and-Wait ARQ

เป็นกลไกที่ง่ายและไม่มีความซับซ้อนมากนัก มีหลักการดังต่อไปนี้

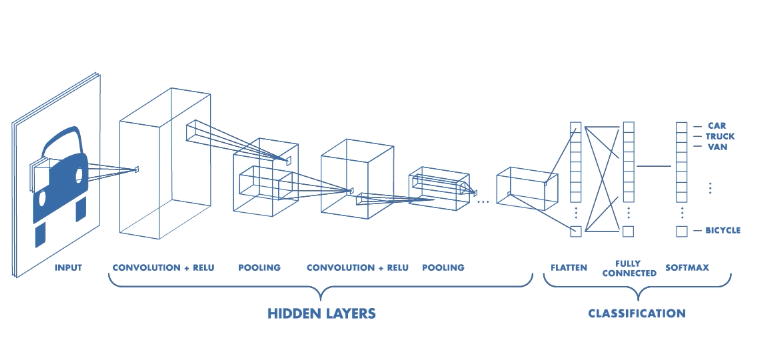
* ผู้ส่งข้อมูลจะต้องทำการสำเนาเฟรมข้อมูลที่จัดส่งไปเอาไว้ก่อนจนกว่าผู้ใช้จะยืนยันว่าได้รับเฟรมข้อมูลนั้นแล้ว เนื่องจากว่าถ้าเฟรมข้อมูลนั้นเกิดการสูญหายหรือเสียในระหว่างการส่ง ผู้ส่งจะสามารถส่งเฟรมข้อมูลไปใหม่ได้
* ในการยืนยันตอบรับเฟรมข้อมูลของผู้รับนั้น ผู้รับจะต้องส่งเฟรม acknowledgment (ACK) มาให้กับผู้ส่ง ซึ่งการส่งเฟรม ACK จะเป็นการบ่งบอกว่าได้รับเฟรมอะไร และเฟรมข้อมูลที่ต้องการถัดไปนั้นคือเฟรมอะไร
* ถ้าผู้รับข้อมูลได้รับเฟรมข้อมูลที่มีความผิดพลาด ผู้รับจะทำการทั้งเฟรมนั้นไป หรือถ้าได้รับเฟรมที่ไม่ต้องการ ก็จะทำการทิ้งเฟรมนั้นไปเช่นกัน
* ผู้ส่งจะใช้ตัวแปร S ในการเก็บข้อมูลว่าได้ส่งเฟรมอะไรออกไป ส่วนผู้รับจะใช้ตัวแปร R ในการเก็บข้อมูลว่าเฟรมถัดไปที่ต้องการคือเฟรมอะไร
* ผู้ส่งจะมีการกำหนดเวลาเอาไว้หลังจากส่งเฟรมข้อมูลออกไปแล้ว ถ้าไม่ได้รับเฟรม ACK กลับมาในเวลาที่กำหนด จะต้องทำการส่งเฟรมข้อมู,นั้นกลับไปอีกครั้งหนึ่ง
* ผู้รับจะส่งเฟรม ACK กลับไปเมื่อได้รับเฟรมข้อมูลที่ไม่มีความผิดพลาดและเป็นเฟรมที่ต้องการ ถ้าผู้รับได้รับเฟรมที่ผิดพลาดหรือเฟรมที่ไม่ต้องการ จะไม่มีการส่งเฟรมใด ๆ ตอบกลับไป



รูปที่ 2.2 การสื่อสารของโปรโตคอล Stop-and-Wait

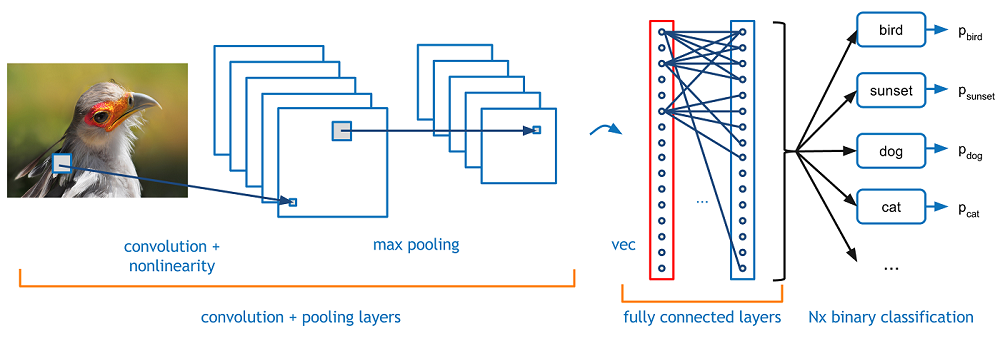
## Machine Learning (ML) for detect images

### Convolutional Neural Network (CNN)



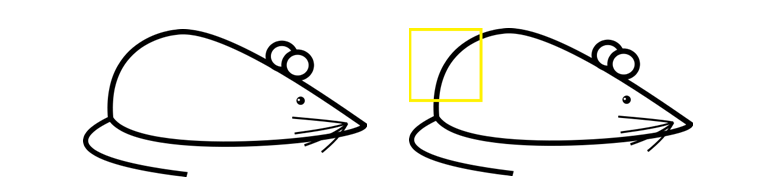
รูปที่ 2.3 แสดงภาพจำลองของ CNN

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

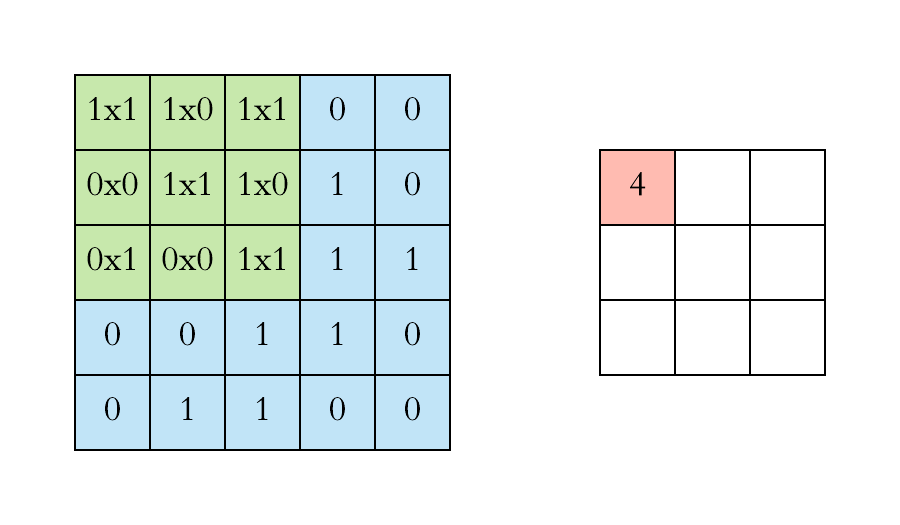


รูปที่ 2.5 แสดงการจำลองจุด focus ของสายตามนุษย์

### Feature Extraction

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

การคำนวณนี้จะเริ่มจากการกำหนดค่าใน ตัวกรอง (filter) หรือ เคอร์เนล (kernel) ที่ช่วยดึงคุณลักษณะที่ใช้ในการรู้จำวัตถุออกโดยปกติตัวกรอง/เคอร์เนลอันหนึ่งจะดึงคุณลักษณะที่สนใจออกมาได้หนึ่งอย่างเราจึงจำเป็นต้องตัวกรองหลายตัวกรองด้วย เพื่อหาคุณลักษณะทางพื้นที่หลายอย่างประกอบกัน

****

รูปที่ 2.6 แสดงการทำ Convolution filter

### Max Pooling

เรามองลองดูหนึ่งในปัญหาของการทำ CNN กันก่อน สมมติเราใช้ CNN ด้วยขนาดตัวกรอง 3x3 พิกเซล แต่เรารู้ดีว่าเวลาเรามองภาพแล้วเราตอบได้ว่ามันคืออะไร เพราะเรามองไปในบริเวณที่กว้างกว่านั้น



รูปที่ 2.7 แสดงการทำ Max Pooling

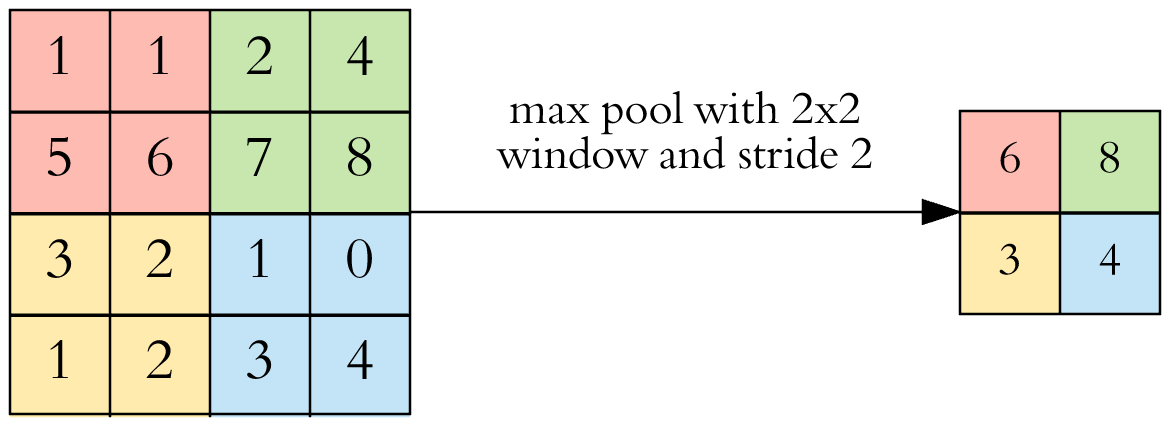
จากภาพ จะเห็นว่าต่อให้รูปภาพมีขนาดสเกลที่เล็กลงแต่เราก็ยังสามารถมองออกว่ามันคือเครื่องปั้นดินเผา แสดงว่าเราจำแนกวัตถุชิ้นนี้ที่ความละเอียดต่ำลง แต่เรากำลังทำ CNN ที่ความละเอียดสูง

Multiscale Analysis จากปัญหาด้านบนเราจะเห็นว่าเป็นไปได้ยากมากหากเราต้องอาศัยข้อมูลที่หยาบหรือละเอียดอย่างใดอย่างหนึ่งในการจำแนกวัตถุดังนั้นในการฝึกเครื่องเราจึงจะเป็นต้องมีข้อมูลทั้งหยาบและละเอียดควบคู่กันไป

ตอนนี้เรารู้แล้วว่าเราจำเป็นต้องคำนวณภาพในหลายสเกล แต่ปัญหาที่สำคัญคือเราจะทำให้การคำนวณอยู่ในรูปหลายสเกลได้อย่างไร หากเราใช้ตัวกรองขนาด 3x3 เรากำลังจะจัดการกับรายละเอียดเล็กๆ (ภาพใหญ่มีรายละเอียดมาก จึงถือว่าเป็นสเกลละเอียด) แต่ด้วยตัวกรองขนาดเท่าเดิม หากทำกับภาพที่ขนาดเล็กลงแล้ว - มันจะครอบคลุมพื้นที่วัตถุเดิมมากขึ้น ดังนั้นถ้าโครงข่ายเราควรจะต้องมีการย่อรูปประกอบด้วย เราก็จะสามารถเข้าถึงความสามารถด้านการวิเคราะห์หลายความละเอียดได้

Pooling คือความสามารถในการย่อรูปแบบหนึ่ง ซึ่งมีสองประเภทหลักที่นิยมกันคือ max pooling และ mean pooling

Max Pooling เป็นตัวกรองแบบหนึ่งที่หาค่าสูงสุดในบริเวณที่ตัวกรองทาบอยู่มาเป็นผลลัพธ์โดยเราจะเตรียมตัวกรองในลักษณะเดียวกับการทำ Feature Extraction ของ CNN มาทาบบนข้อมูลแล้วเลือกค่าที่สูงที่สุดบนตัวกรองนั้นมาเป็นผลลัพธ์ใหม่ และจะเลื่อนตัวกรองไปตาม Stride ที่กำหนดไว้ โดยขนาดตัวกรองของการทำ max pooling จะนิยมเรียกกันว่า pool size

****

รูปที่ 2.8 แสดงการทำ max pool with 2x2

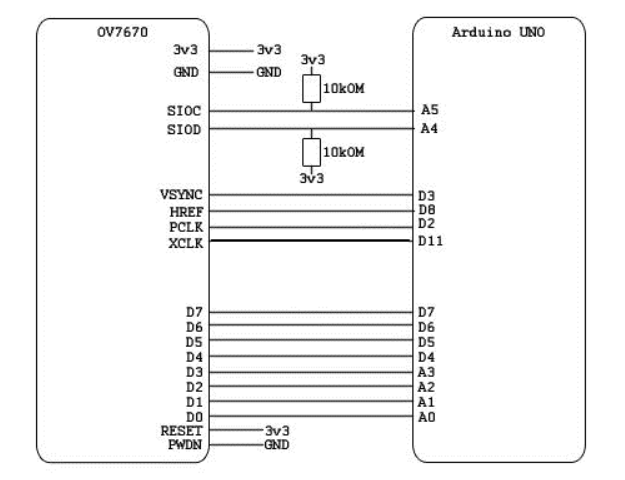
## การติดตั้งกล้อง OV7670 Camera Module

ขั้นตอนการใช้งาน OV7670 (โมดูลกล้อง) ใช้สำหรับการทดสอบว่ากล้องสามารถทำงานได้หรือไม่

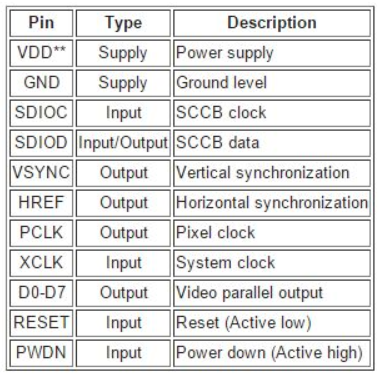
1. ดาวโหลดโฟลเดอร์ OV7670 DataCom จาก https://drive.google.com/drive/folders/

0B8uPgCHycI9VLXdvUWlxTi02ZlU?usp=sharing

1. ต่อโมดูลกล้องเข้ากับ Arduino UNO ดังภาพ โดย XCLK ไม่ต้องต่อตัวต้านทาน

****

รูปที่ 2.9 การเชื่อมต่อโมดูลกล้อง OV7670 เข้ากับ Arduino UNO

****

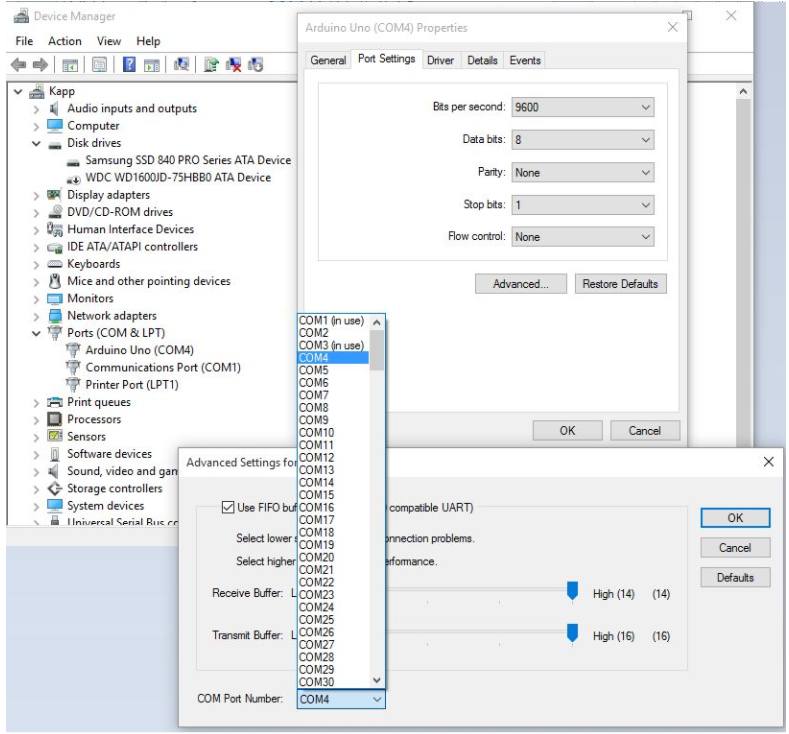
รูปที่ 2.10 คำอธิบายขาพินของโมดูลกล้อง OV7670

1. เปิดโฟลเดอร์ Camera ในโฟลเดอร์ OV7670 DataCom จากนั้นอัพโหลดโปรแกรม Camera.ino เข้าไปใน Arduino UNO
2. หากต่อทุกอย่างถูกต้องและ Upload ได้ถูกต้องสามารถดูภาพได้จากโปรแกรม ReadSerialPort.exe ได้ โดยเลือก comport ให้ถูกต้อง ให้กดปุ่ม start

****

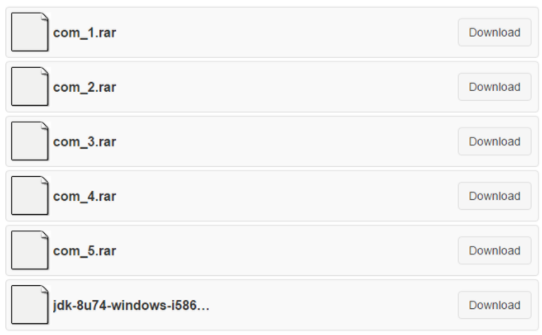
รูปที่ 2.11 ตัวอย่างการรับภาพจากโมดูลกล้อง OV7670

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



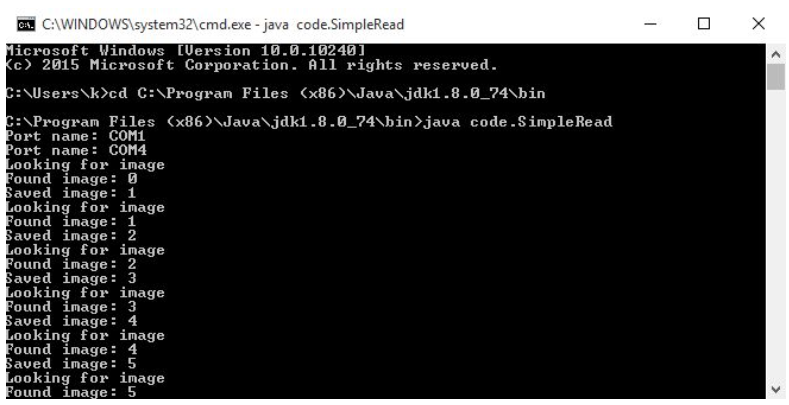
รูปที่ 2.12 การเปลี่ยนแปลงค่า Comport

1. สร้าง โฟลเดอร์ชื่อ out ใน Drive C
2. ทำการติดตั้งโปรแกรม jdk-8u74-windows-i68 (มีให้ใน OV7670 DataCom)
3. Extract com\_X.rar (โดย X คือเลข Comport ที่ได้ตั้งไว้ในขั้นตอนที่ 4)



รูปที่ 2.13 การ extract ไฟล์ที่ได้รับมา

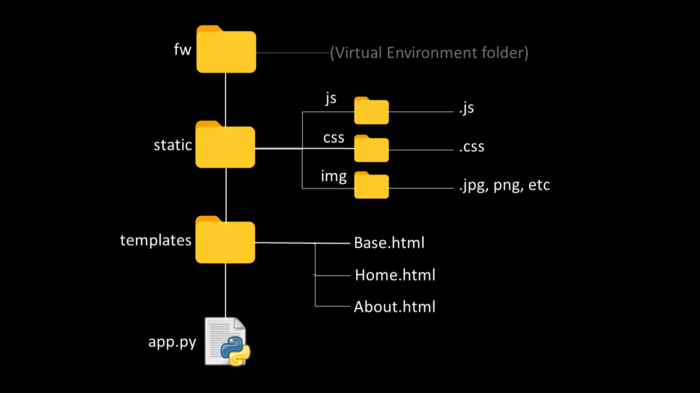
1. Copy โฟลเดอร์ code ที่อยู่ใน com\_X และวางใน C:\Program Files (x86)\Java\jdk1.8.0\_74\bin
2. Copy file win32com.dll และวางใน C:\Program Files (x86)\Java\jdk1.8.0\_74\jre\lib
3. Copy file comm.jar ในโฟลเดอร์ lib และวางใน C:\Program Files (x86)\Java\jdk1.8.0\_74\jre\lib\ext
4. Copy file javax.comm.properties ในโฟลเดอร์ lib และวางใน C:\Program Files (x86)\Java\jdk1.8.0\_74\jre\lib
5. เปิด cmd พิมพ์ cd C:\Program Files (x86)\Java\jdk1.8.0\_74\bin
6. พิมพ์ java code.SimpleRead
7. หากทุกอย่างถูกต้อง cmd จะขึ้นข้อความดังภาพ และ สามารถดูรูปภาพใน C:\out



รูปที่ 2.14 ตัวอย่างการแสดงผลเมื่อตั้งค่าถูกต้องบน cmd

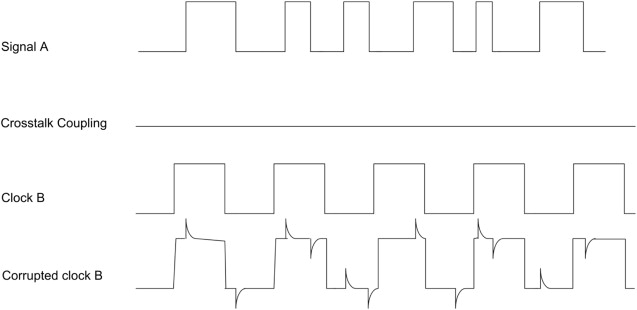
## การสร้าง Server ด้วย python flask

รูปที่ 2.15 Logo Flask

1. Flask คือ web framework ที่เขียนขึ้นมาสำหรับ Pyth[on](https://saixiii.com/python-programming/) เพื่อใช้ร่วมกัน webserver โดย flask ถูกเรียกว่า micro framework เพราะว่า เป็นระบบไม่ต้องการเครื่องมือ หรือ library อะไรเพิ่มเติม สามารถทำงานด้วยตัวเองได้
2. ศึกษาการ Stream ค่า โดยใช้ flask โดยนำมาใช้หรับการ Stream ค่าจาก Serial port ไปยังหน้า Website แบบ Realtime
3. ศึกษาการสร้าง Route ไว้ใช้ในการ ส่งค่าจาก Web Client มาจาก Server โดยเป็นคำสั่ง ที่ต้องการส่งไปยัง Arduino
   1. Route คือ class ที่ใช้จัดการกับเส้นทาง เข้า-ออก ของเว็บไซต์เรา พูดให้เข้าใจง่ายๆคือ เป็นตัวกำหนด Path ว่า Path นี้ให้ทำงานที่ Controller ไหน หรือ แสดง View ไหน รวมถึงสามารถระบุการส่ง Parameter ผ่านเจ้าตัว Route ได้ด้วย
4. โครงสร้างของ Flask 

รูปที่ 2.16 แสดงโครงสร้างของ Flask

## ปัญหาครอสทอล์ค (Crosstalk)



รูปที่ 2.17 แสดงการเกิด Noise ในขา Corrupted clock B

การแทรกสัญญาณข้าม หรือ สัญญาณแทรกข้าม (อังกฤษ: crosstalk (XT)) หมายถึงสัญญาณไฟฟ้าข้ามไปรบกวนกัน เป็นปรากฏการณ์ที่เกิดขึ้นได้ในการส่งสัญญาณในขณะที่สัญญาณเดินทางไปในสื่อสายทองแดงด้วยความเร็วสูง ปรากฏการณ์นี้มักพบได้เมื่อสื่อที่ใช้ในการส่งสัญญาณเป็นสายทองแดงตั้งแต่สองคู่ขึ้นไป เช่น โทรศัพท์ ซึ่งมีสายทองแดงมัดรวมกันเป็นคู่ นอกจากนี้ปรากฏการณ์นี้ยังสามารถเกิดจากสนามแม่เหล็กภายในสายทองแดงเองขณะกำลังส่งสัญญาณ ทำให้เกิดสัญญาณเสียงข้ามไปยังคู่สนทนาคู่อื่นได้แม้ว่าจะมีการหุ้มแผ่นโลหะไว้เพื่อป้องกันสัญญาณรบกวนแล้วก็ตาม

ปัญหาหนึ่งของการใช้สายส่งสัญญาณ หรือ ครอสทอล์ค (Crosstalk) ซึ่งสัญญาณหนึ่งจะรบกวนสัญญาณบนสายอีกสายหนึ่ง สายคู่เกลียวได้ถูกออกแบบมาเพื่อลดปัญหานี้โดยการบิดคู่สายเป็นเกลียวทำให้สัญญาณรบกวนในแต่ละสายหักล้างกัน ถ้าจำนวนเกลียวต่อหน่วยความยาวยิ่งมากเท่าใด จะทำให้ป้องกันคอรสทอล์คได้ดีแต่ข้อเสียคือจะทำให้ความยาวของสายเพิ่มขึ้น

ในกรณีของอุปกรณ์โทรคมนาคมที่เชื่อมต่อด้วยสายไฟจริง crosstalk จะเกิดขึ้นเมื่อใดก็ตามที่สายเหล่านี้ข้ามกัน สิ่งนี้ทำให้เกิดความไม่สงบในสัญญาณซึ่งปรากฏอยู่ในหูของผู้ฟังว่าเป็นสัญญาณเสียงหรือชิ้นส่วนของคำพูด การหุ้มฉนวนสายไฟและการบิดอย่างแน่นหนาสามารถลดสิ่งนี้ได้อย่างมากแม้ว่าการเดินสายจะเสื่อมสภาพในที่สุดเมื่ออายุมากขึ้นซึ่งจำเป็นต้องซ่อมแซม หากผู้บริโภคประสบ crosstalk บนอุปกรณ์เช่นโทรศัพท์ควรรายงานไปยัง บริษัท โทรศัพท์เพื่อให้พวกเขาสามารถแก้ไขปัญหาได้

# บทที่ 3 ขั้นตอนและวิธีการดำเนินงาน

## ขั้นตอนการดำเนินงาน

1. วางแผนและแบ่งหน้าที่รับผิดชอบของสมาชิกแต่ละคนในกลุ่ม
2. ศึกษาและรวบรวมข้อมูลที่ใช้ในการทำระบบต่าง ๆ และความต้องการของระบบ
3. เริ่มออกแบบระบบการทำงานเบื้องต้น นำมาประกอบเข้าด้วยกันเพื่อดูภาพรวม
   1. ออกแบบการทำงาน Protocol ในการสื่อสารรับส่งข้อมูลและ Frame Design
   2. ออกแบบรูปแบบการรับส่งข้อมูล (Flow Control)
   3. ออกแบบรูปแบบการตรวจสอบข้อมูล (Error Control)
   4. ออกแบบการแปลงคลื่นสัญญาน (FM)
   5. ออกแบบการทำงานของระบบควบคุม Arduino
4. ทำงานในส่วนที่แต่ละคนได้รับมอบหมาย
5. ทดสอบการทำงานของแต่ละส่วนและแก้ไขปัญหา เมื่อแต่ละส่วนทำงานถูกต้องแล้วจึงนำแต่ละส่วนมารวมกัน
6. ทดสอบระบบรวมทั้งหมดและแก้ไขปัญหาที่เกิดขึ้น
7. นำปัญหาที่พบเจอ มาจดบันทึก และระบุแนวทางแก้ไขเพื่อเป็นประโยชน์ในครั้งถัด ๆ ไป
8. จัดทำเอกสารรายงานรูปเล่ม

## **ระยะเวลาดำเนินงาน**

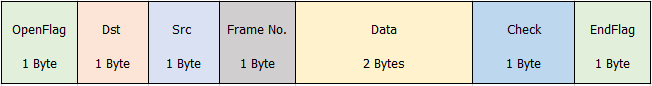
วันที่ 5 พฤศจิกายน 2563 ถึง 25 พฤศจิกายน 2563

## การออกแบบระบบ

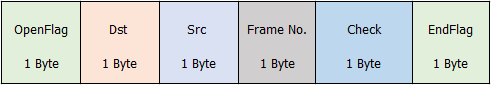
การออกแบบระบบการถ่ายภาพและส่งข้อมูลผ่านคลื่น FM สามารถแบ่งเป็นหัวข้อได้ ดังนี้

1. การสื่อสารระหว่างอุปกรณ์
2. การติดตั้ง และการทำงานของกล้อง (Camera)
3. ประมวลผลภาพด้วย Machine Learning (ML)
4. การควบคุมคำสั่งต่างๆ
5. Flowchart การทำงาน
6. การสื่อสารระหว่างอุปกรณ์
   1. เฟรมข้อมูล

* 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



รูปที่ 3.1 Frame Design ของ Data Frame



รูปที่ 3.2 Frame Design ของ ACK Frame

* 1. Flow Control

เลือกใช้เป็น Stop-and-Wait ARQ

* + 1. ผู้ส่งข้อมูลจะต้องทำการสร้างเฟรมและสำเนาเฟรมข้อมูลที่จัดส่งไปเอาไว้ก่อนจนกว่าผู้ใช้จะยืนยันว่าได้รับเฟรมข้อมูลนั้นแล้ว
    2. ผู้รับจะส่งเฟรม acknowledgment (ACK) มาให้กับผู้ส่งถ้าได้รับเฟรมที่ถูกต้อง โดยจะบอกว่าเฟรมถัดไปที่ต้องการรับมีเลขลำดับเท่าใด
    3. ถ้าผู้รับข้อมูลได้รับเฟรมข้อมูลที่มีความผิดพลาด หรือไม่ต้องการ เฟรมที่รับมานั้นจะถูกทิ้งไป
    4. ผู้ส่งจะมีการกำหนดเวลาเอาไว้หลังจากส่งเฟรมข้อมูลออกไปแล้ว ถ้าไม่ได้รับเฟรม ACK กลับมาในเวลาที่กำหนด ผู้ส่งจะทำการส่งเฟรมข้อมูลนั้นกลับไปอีกครั้งหนึ่ง

ฝั่งส่งจะทำการส่งข้อมูลทีละเฟรมและรอการตอบรับด้วยเฟรม ACK จากฝั่งรับโดยมี timeout ที่ 1500 มิลลิวินาที หากไม่มีการตอบรับกลับมาหรือเฟรม ACK มีความผิดพลาด ฝั่งส่งจะทำการวนส่งเฟรมให้ใหม่อีกครั้งแล้วทำการรอซ้ำต่อไปจนกว่าเฟรม ACK ที่ได้รับจะถูกต้องแล้วจึงเริ่มส่งเฟรมต่อไปดังส่วนของโค้ดต่อไปนี้



รูปที่ 3.3 ส่วนของโค้ดในการส่งข้อมูลด้วยโปรโตคอล Stop-and-Wait ARQ

ในส่วนของฝั่งรับ จะทำการรับเฟรมข้อมูลเข้ามาแล้วตรวจสอบความถูกต้องของเฟรมข้อมูล ถ้าเฟรมที่ได้รับถูกต้อง จะทำการรับเฟรมมาเลือกนำส่วนที่เป็นข้อมูลไปเก็บไว้และส่ง ACK กลับไปให้ฝั่งส่ง เมื่อได้รับเฟรมครบแล้วดังสังเกตได้จากว่า ENDFLAG เป็น “0” จะจบการรับข้อมูล ดังส่วนของโค้ดต่อไปนี้



รูปที่ 3.4 ส่วนของโค้ดในการรับข้อมูลด้วยโปรโตคอล Stop-and-Wait ARQ

* 1. Error Detection and Error Control

เลือกใช้วิธีการตรวจจับข้อผิดพลาดโดยวิธี Checksum แล้วนำค่ามา mod 2 โดยจะเริ่มนับ Checksum ตั้งแต่ open flag ไปจนถึง data 2 ไบต์ในกรณีของเฟรมข้อมูลและนับจาก open flag ไปถึง frame number ในกรณีของเฟรม ACK และมีการควบคุมแก้ไขข้อผิดพลาดโดยการส่งเฟรมข้อมูลที่ผิดพลาดไปใหม่อีกครั้งหนึ่งโดยมีโค้ดที่ใช้ในการตรวจสอบความถูกต้องดังนี้



รูปที่ 3.5 ส่วนของโค้ดในการตรวจสอบความถูกต้องของข้อมูล

* 1. รูปแบบการส่งข้อมูล

ใช้รูปแบบการส่งข้อมูลผ่านคลื่นสัญญาณแบบ FM เพื่อสื่อสารระหว่าง PC1 และ PC2

* 1. การเข้ารหัสและการถอดรหัสข้อมูล

สัญญาณ Digital ใช้การเข้ารหัสและถอดรหัสข้อมูลแบบ 2-FSK และสัญญาณ Analog ใช้การเข้ารหัสและถอดรหัสข้อมูลแบบ 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

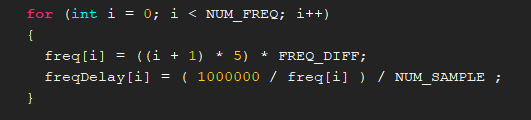
0 = 5 cycles / baud

1 = 10 cycles / baud

1.6.1.2. Partition = 1 bit / baud

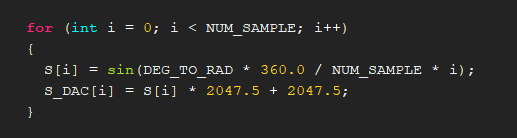
1.6.1.3. Baud rate = 100 baud

1.6.1.4. Bit rate = 100 bps



รูปที่ 3.6 ส่วนของโค้ดที่เป็นการกำหนดความถี่และเวลาการส่งคลื่น

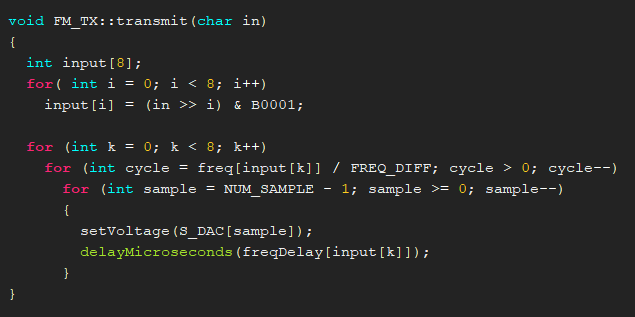
1.6.1.5. คลื่น Digital ทำการ sampling สัญญาณออกเป็น 4 sample จากมุม 4 องศา คือ 0, 90, 180 และ 270 องศา



รูปที่ 3.7 ส่วนของโค้ดที่ทำการ sampling รูปแบบการส่งคลื่นออกเป็น 4 ส่วน

1.6.2. การแปลงข้อมูลเป็นข้อมูล Digital และการเข้ารหัสข้อมูลแบบ 2-FSK

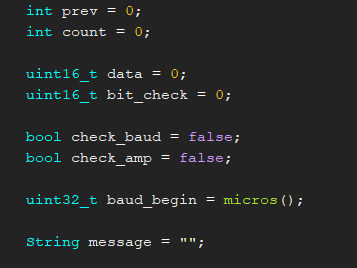
ก่อนการส่งข้อมูลจะนำข้อมูล Analog ที่จะทำการส่งมาทำการแปลงเป็นข้อมูล Digital โดยการเริ่มเข้ารหัสและส่งข้อมูล จาก LSB ไป MSB ของข้อมูลแต่ละตัวอักษร นำข้อมูลที่ได้มาเข้ารหัสตาม Encoding rule แล้วทำการส่งคลื่นออกไป ตามลักษณะของคลื่นที่ได้จากการ sampling คลื่นที่ส่งออกไปนั้นจะเป็นการส่งคลื่นที่มีขนาด 8 baud / byte เท่ากับ 1 ตัวอักษร



รูปที่ 3.8 ส่วนของโค้ดที่ทำการส่งคลื่น digital ออกไป 1 ตัวอักษร

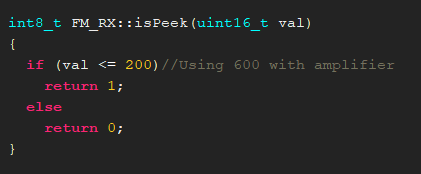
1.7. 2-FSK Demodulation

การ demodulation เป็นการรับข้อมูลสัญญาณ FM ที่ได้จากเสาสัญญาณที่ผ่าน amplifier มาทำการถอดรหัสออกเป็นข้อมูล

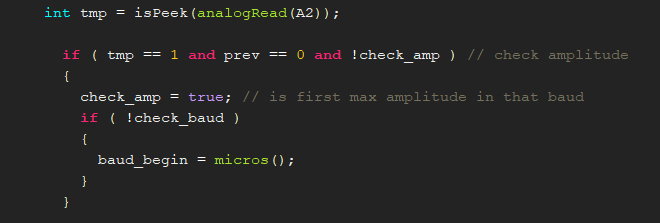


รูปที่ 3.9 ส่วนของโค้ดที่เป็นตัวแปรสำคัญในการถอดรหัส

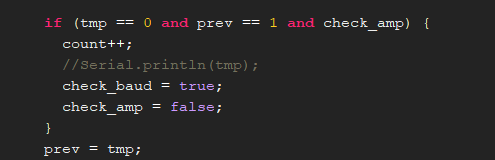
1.7.1. เมื่อมีสัญญาณ analog เข้ามาจะรับเข้าผ่านขา A2 ของ adruino UNO3 แล้วนำข้อมูลที่ได้มาทำการเช็คว่าค่าที่รับเข้ามาเป็นค่า V peek ของคลื่น และค่าที่รับมาก่อนหน้ายังไม่อยู่ในช่วง V peek ของคลื่น แสดงว่าเป็นการเจอคลื่นลูกนี้เป็นครั้งแรก และทำการนับจำนวนลูกคลื่นนั้น



รูปที่ 3.10 ส่วนของโค้ดที่การเช็คว่าคลื่นที่รับเข้ามาอยู่ในช่วงไหน

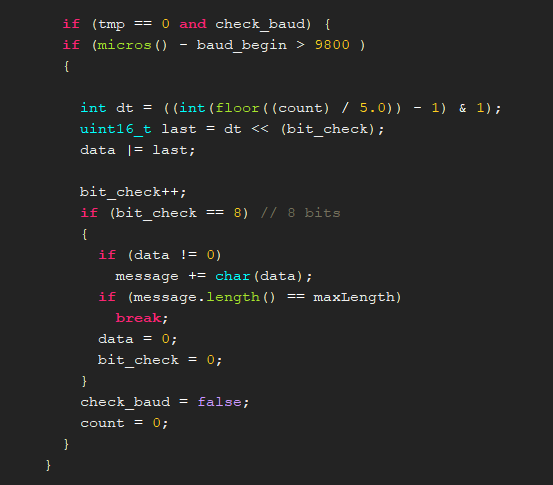


รูปที่ 3.11 ส่วนของโค้ดที่เป็นการเช็คการเจอคลื่นครั้งแรกและการเริ่มต้นของแต่ละ baud



รูปที่ 3.12 ส่วนของโค้ดที่ทำการนับลูกคลื่นเมื่อมีการเจอลูกคลื่นครั้งแรก

1.7.2. เมื่อค่าของคลื่นที่รับมาเป็นค่าที่ไม่ได้อยู่ในช่วง V peek และมีการเช็คว่ามีการเริ่ม baud นั้นมาแล้วจนครบช่วงเวลาของแต่ละ baud จะนำข้อมูลของลูกคลื่นที่นับได้ใน baud นั้น มาทำการถอดรหัส ตาม encoding rule แล้วทำการ shift bit จาก LSB ไป MSB และเมื่อข้อมูลที่ได้ครบ 8 bits แสดงว่าได้ข้อมูลมา 1 ตัวอักษร แล้วนำตัวอักษรไปเก็บไว้ในข้อมูลที่เป็นข้อความ



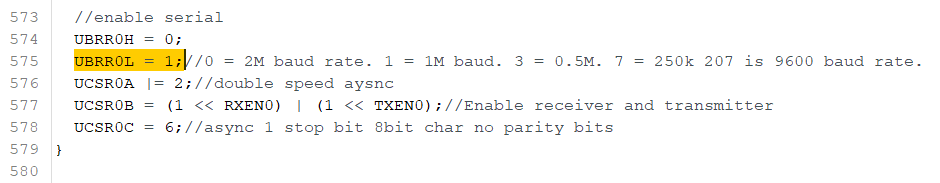
รูปที่ 3.13 ส่วนของโค้ดที่เป็นการถอดรหัสข้อมูล

1. การติดตั้ง และการทำงานของกล้อง (Camera)
   1. การติดตั้ง
      1. ประกอบฮาร์ดแวร์ของกล้องและเชื่อมต่อสายไฟ
      2. ทดสอบอัพโหลดโค้ต Arduino (จากไฟล์ Arduino3/Camera/Camera.ino) ที่มีการแก้ไข ลดขนาดไฟล์รูปถ่ายให้เล็กลงจาก 320x240px ลดลงเหลือ 160x120px เนื่องจากรูปที่ได้จะพอดีกับเฟรมที่ต้องการใช้การในการประมวลผล เพิ่มความเร็วและประสิทธิภาพในการถ่ายรูป



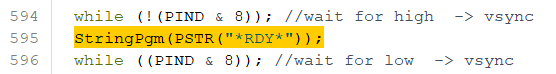
รูปที่ 3.14 ตัวอย่างภาพที่ถ่ายได้จากกล้อง (160x120px)

* + 1. Arduino ที่ต่อกับ Raspberry pi เชื่อมต่อผ่าน Serial port 1,000,000 Baud

รูปที่ 2.1.3 แสดงการตั้งค่า Buad ที่เชื่อมต่อของ Arduino 

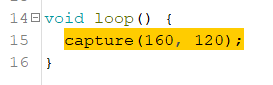
(จากไฟล์ Arduino3/Camera/CameraController.cpp line 575)

* + 1. เมื่อ Arduino เริ่มทำงานจะทำการถ่ายรูปไปเรื่อย ๆ และส่งข้อมูลมาทีละ Byte ผ่านทาง Serial port โดยจะมี flag เปิดเป็นคำว่า “\*RDY\*” สามารถเปลี่ยนได้จากในโค้ต

รูปที่ 3.15 แสดงการตั้งค่า flag เปิดของรูปภาพ

(จากไฟล์ Arduino3/Camera/CameraController.cpp line 595)

* + 1. ตั้งค่าขนาดรูปภาพที่ต้องการถ่าย ซึ่งในที่นี้ตั้งค่าเป็น 160x120px

รูปที่ 3.16 แสดงการตั้งค่าขนาดของรูปภาพ

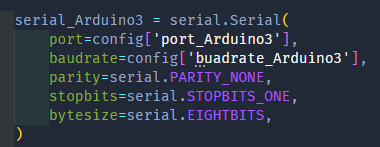
(จากไฟล์ Arduino3/Camera/Camera.ino line 15)

* + 1. เขียนโปรแกรมเพื่อเชื่อมต่อ Serial port ด้วยภาษา Python โดยใช้ Library “serial”

รูปที่ 3.17 เรียกใช้ library serial

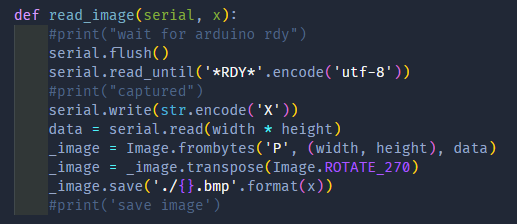
(จากไฟล์ PC2/server.py line 7)

* + 1. เปิดการเชื่อมต่อกับ Serial port และตั้งค่า Port, baudrate (Setting ต่าง ๆ จะถูกเรียกใช้งานจากไฟล์ Config (PC2/config.json)

รูปที่ 3.18 ตั้งค่าการเชื่อมต่อ Serial

(จากไฟล์ PC2/server.py line 186)

* + 1. อ่าน Data ที่ส่งผ่าน Serial port จนกว่าจะเจอคำว่า “\*RDY\*” โดยต้อง decode ข้อมูลที่ได้เนื่องจากการส่งข้อมูลผ่าน Serial นั้นเป็นจะถูกแปลงข้อมูลเป็น byte
       1. เมื่อเจอคำว่า \*RDY\* แล้วให้อ่านข้อมูลอีก 160x120 Byte (ข้อมูลของรูปภาพ)
       2. แปลงข้อมูลที่ได้เป็นรูปภาพ โดยใช้ Image จาก Library PIL
       3. บันทึกรูปภาพที่ได้ เพื่อรอนำไป Process ต่อไป

รูปที่ 3.19 การบันทึกรูปภาพจาก Data

(จากไฟล์ PC2/Image\_processing.py line 83)

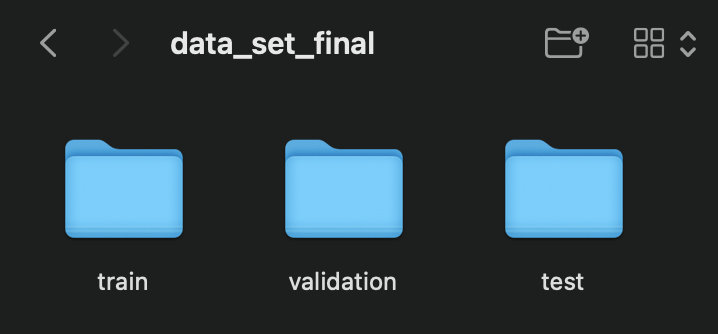
* 1. การทำงานของกล้อง [เขียนเพิ่ม]

1. 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

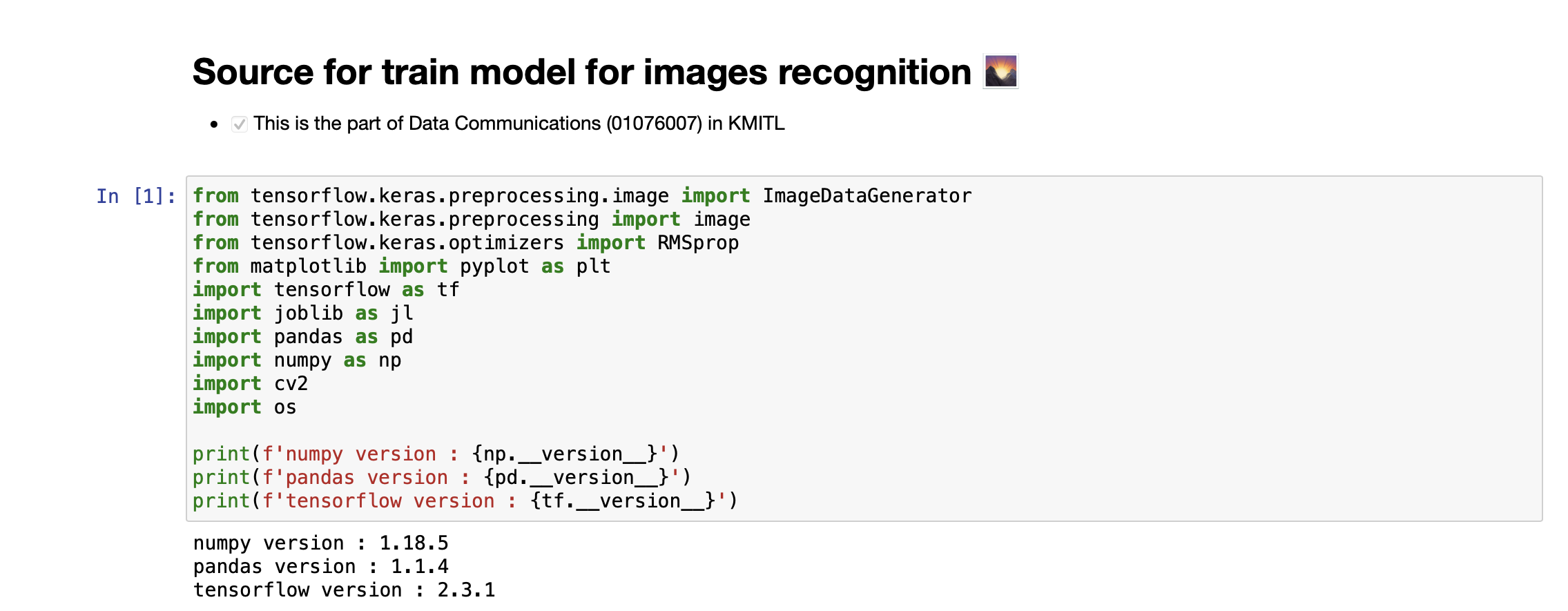
3.2 แต่ละ folder จะแบ่งออกเป็น 16 folder ตาม class ที่ต้องการจำแนก



รูปที่ 3.21 การแบ่งโฟลเดอร์ย่อยตาม class ที่ต้องการจำแนก

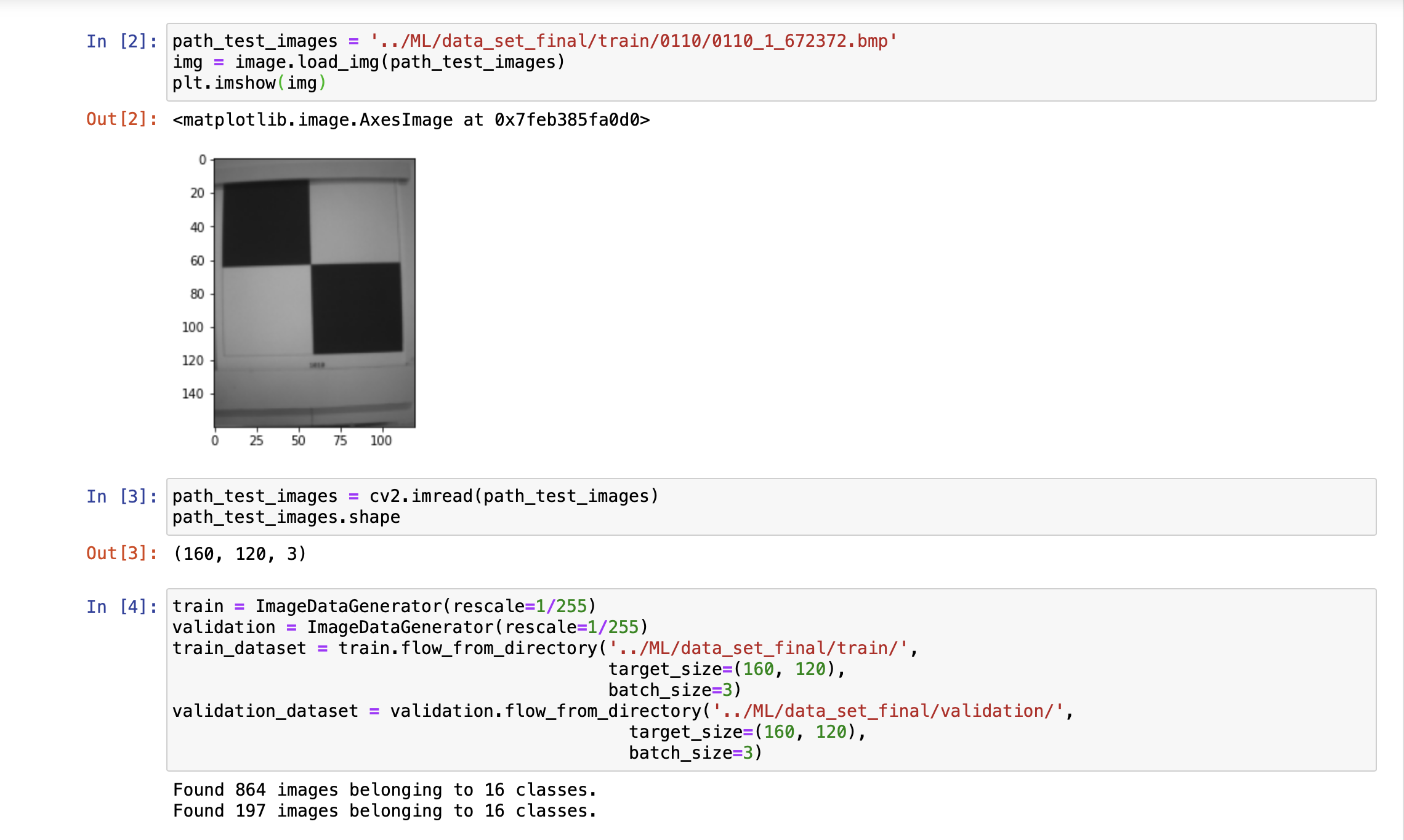
รายละเอียดวิธีการประมวลผลภาพด้วย Machine Learning (ML) มีดังนี้

* 1. Import library ที่จะใช้ (ดังรูปที่ 3.5)



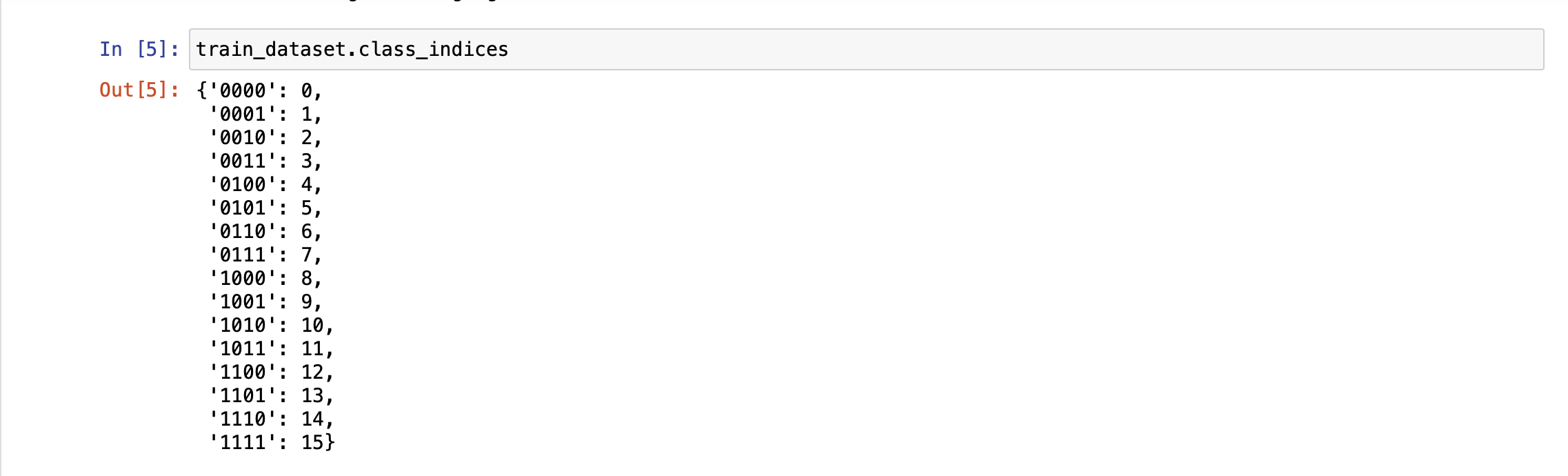
รูปที่ 3.22 การ import library

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



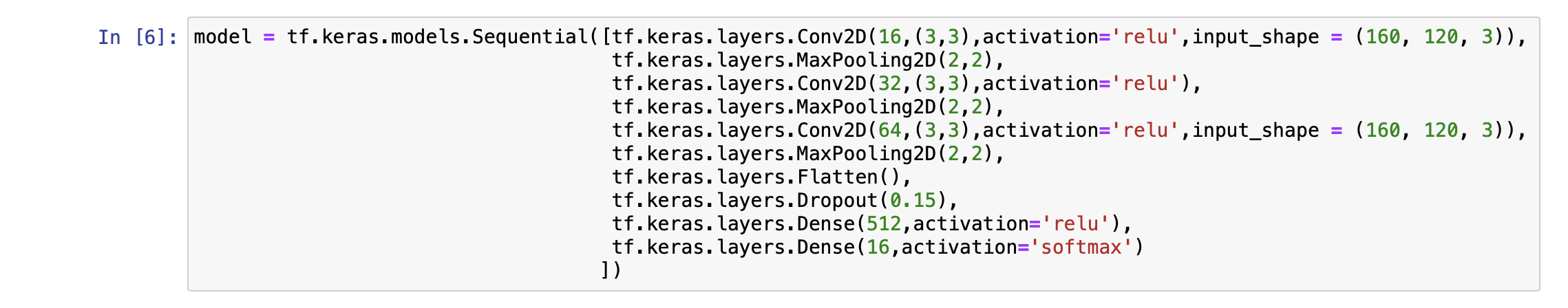
รูปที่ 3.23 การทดลองนำเข้า, แสดงผลรูปภาพ และ import ชุดข้อมูลต่าง ๆ

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



รูปที่ 3.24 แสดงผล class ต่าง ๆ ตามชุดข้อมูล

* 1. สร้าง model ตามที่ได้ออกแบบไว้ (ดังรูปที่ 3.8)



รูปที่ 3.25 การสร้าง model

* 1. Compile และ Train model (ดังรูปที่ 3.9)



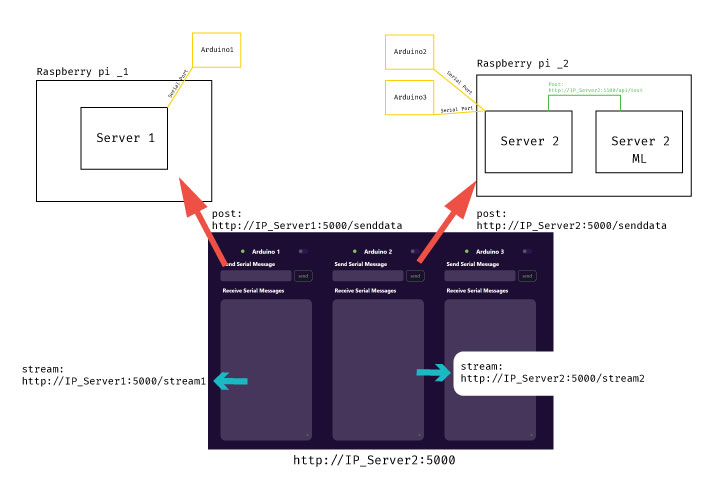
รูปที่ 3.26 การ compile และ train model

* 1. Save model (ดังรูปที่ 3.10)

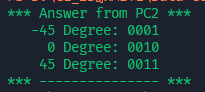


รูปที่ 3.27 การบันทึก model ที่ได้รับ

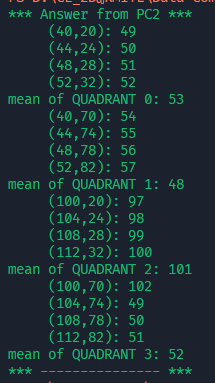
1. การควบคุมคำสั่งต่างๆ
   1. การทำงานของระบบ การออกแบบระบบต้องการให้มีความสเถียรและความเร็วสูงสุด จึงทำเอาการเขียนแบบ Threading เข้ามาเป็นส่วนการทำงานหลัก เพื่อให้โปรแกรมสามารถทำงานทุกส่วนพร้อมกันได้ และใช้ภาษา Python เป็นหลัก เนื่องจากมีความยืดหยุ่นในการเขียนและมี library ที่ช่วยให้การทำงานได้ง่ายขึ้น
      1. Raspberry PI 1 (PC1) เชื่อมต่อกับ arduino 1 มี Thread การทำงานหลัก ๆ 3 thread โดยการเขียนโปรแกรมจะเน้นไปทางการใช้ตัวแปร global เป็นหลัก เพื่อให้ทุก function ในแต่ละ thread สามารถใช้ข้อมูลร่วมกันได้
         1. Thread ที่ 1 ทำหน้าที่คุม Web server ได้แก่
            1. Post /send data ทำหน้าที่ส่งข้อความไปยัง Serial port ของ arduino 1
            2. Stream /stream1 ทำหน้าที่ stream data ที่ส่งมายัง PC ไปยังหน้าเว็บ
         2. Thread ที่ 2 ทำหน้าที่รับ input จาก keyboard แล้วส่งข้อความที่ได้ยังไป arduino ผ่านทาง serial port
         3. Thread ที่ 3 ควบคุมการเชื่อมต่อของ arduino โดยทำหน้าที่ควบคุมทั้ง read และ write

รูปที่ 3.28 การเชื่อมต่อของระบบ

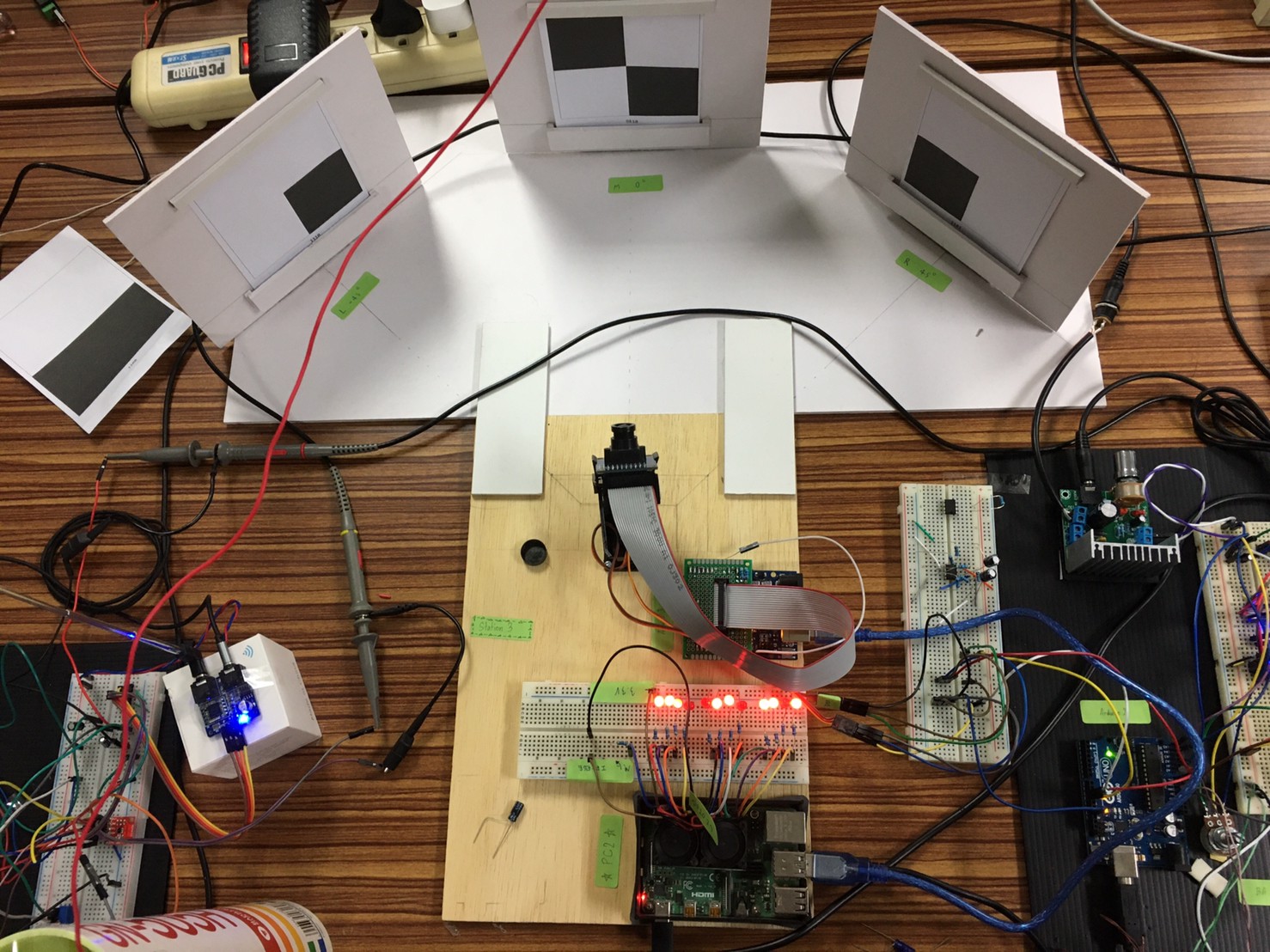
* 1. Raspberry PI 2 เชื่อมต่อกับ arduino 2 และ arduino 3 โดยใช้หลักการเดียวกันกับ Raspberry PI 1 ที่นำเอา thread มาใช้ในการทำงาน
     1. Server ที่ทำหน้าที่ประมวลผลภาพ (ML) จะเปิดเป็น server แยก ที่ port 5500 (<http://ip_server:5500/api/test>) เหตุผลที่ต้องแยกออกมาเป็นอีก server เพราะว่าจะสามารถสร้างความยืดหยุ่นให้ระบบได้มากกว่า หาก raspberry pi ตัวที่ 2 ไม่สามารถประมวลผลทั้งหมดได้ จะสามารถย้ายเอา server ไปรันไว้ที่อื่นแล้วยิง request ไปที่ server นั้นแทนได้
     2. Server หลัก มี 4 thread
        1. Thread 1 ทำหน้าที่ควบคุม web server
           1. โดยเมื่อเข้ามายัง port 5000 ใน home directory (<http://ip_server:5000/>) server จะตอบกลับเป็น static ไฟล์ของหน้าเว็บ (main.html)
           2. /senddata ทำหน้าที่ส่งข้อความไปยัง serial port ของ arduino 2
           3. /stream2 ทำหน้าที่ stream ข้อความที่ได้จาก arduino 2 ไปยังหน้าเว็บไซค์
        2. Thread 2 ทำหน้าที่ควบคุมการเชื่อมต่อของ raspberry pi และ arduino 2 ควบคุมทั้ง read และ write
        3. Thread 3 ควบคุมการ display Led แสดงข้อมูล binary ที่กล้องสามารถ detect ได้ 12bit แบ่งเป็น left 4 bit, middle 4 bit, right 4 bit
        4. Thread 4 ทำหน้าที่ควบคุมการรับคำสั่งให้กล้องหมุนและถ่ายรูป โดยมีสองแบบ คือถ่ายทั้ง 3 มุม และถ่ายเฉพาะมุม
           1. หมุนถ่ายทั้ง 3 มุม สิ่งที่ได้จาก function จะเป็น [left, middle, right] รหัสเลขฐาน 10 ของภาพทั้ง 3 มุม โดยกล้องจะเริ่มจากการหมุนไปฝั่งซ้าย แล้วทำการถ่ายรูป โดยจะมีการ flush รูปทิ้ง 3 รูป (จากการทดสอบแล้ว 3 รูปแรกมักเป็นรูปที่ถ่ายแล้วใช้ไม่ได้) แล้วทำการถ่ายรูปที่ 4 เพื่อนำมาประมวลผล delay 0.5 วินาที แล้วทำซ้ำอีกครั้ง นำคำตอบทั้งสองครั้งมาตรวจสอบ หากเป็นคำตอบเดียวกัน ให้หมุนไปยังอีกสองมุม แต่หากไม่เหมือนกัน จะทำการถ่ายซ้ำ จนกว่าจะได้คำตอบตรงกัน
           2. หมุนไปถ่ายยังมุมที่ระบบ สิ่งที่ได้จาก function จะเป็นค่าสี 16 จุด และอีก 4 ค่าเป็นค่าเฉลี่ย
  2. คำสั่งต่าง ๆ
     1. capture หาก server 2 ได้รับคำสั่งนี้ จะทำการถ่ายภาพ ทั้ง 3 มุม โดยการสั่งงานนั้นสามารถทำได้ 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

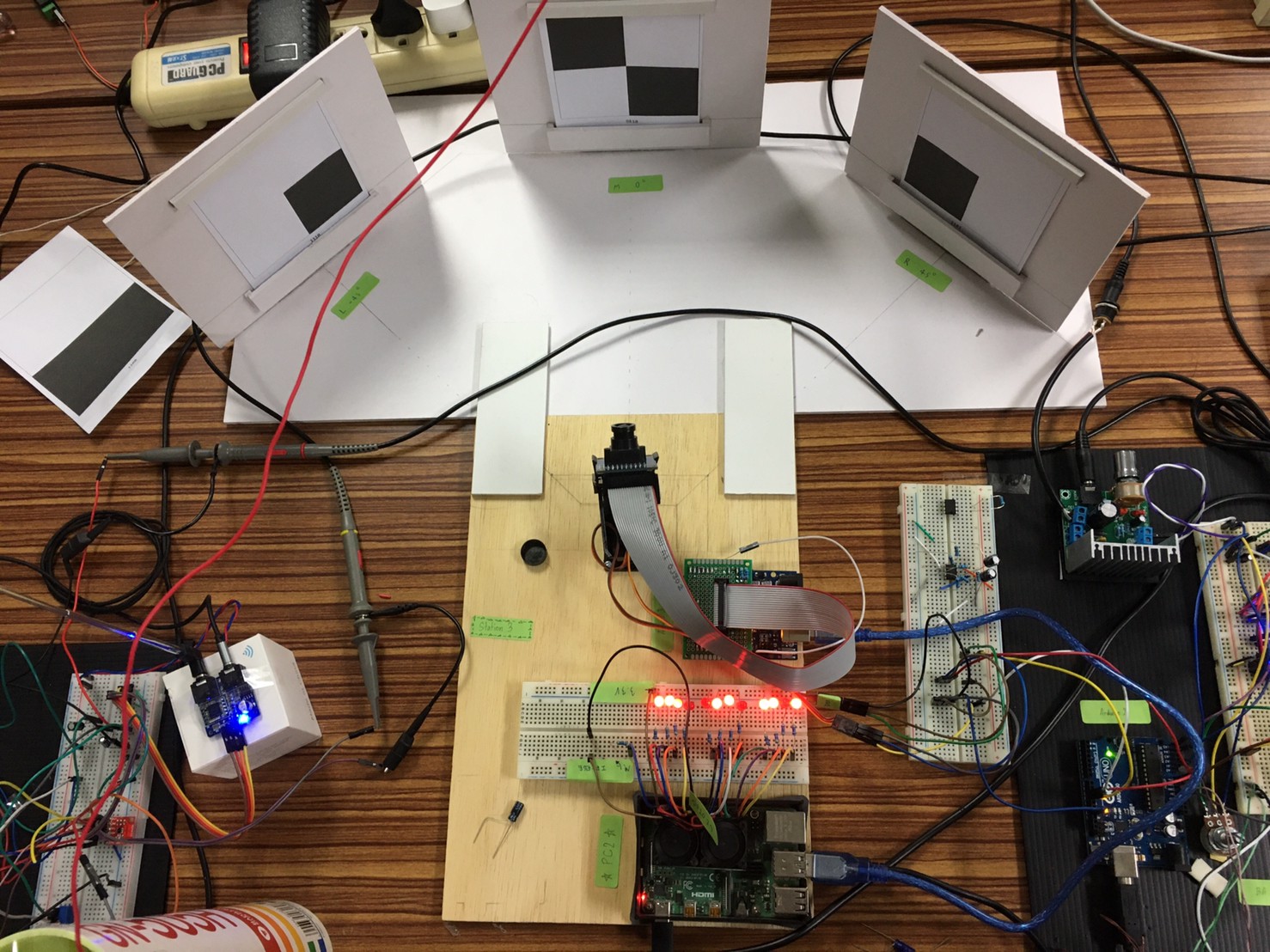
รูปที่ 3.29 แสดงการ decode ข้อความ “X123” ในฝั่ง PC1

* + - 1. การสั่งจาก arduino หากพิมพ์คำว่า capture arduino 1 จะทำการส่งข้อความนั้นมายัง arduino 2 จะทำการส่งข้อความที่ได้ไปยัง serial , server จะสั่งให้กล้องหมุนและบันทึกรูปภาพตามระบบต่อไป
      2. หากสั่งจากหน้าเว็บไซต์ <http://ip_server2:5000/> (ใช้สำหรับการ debug) พิมพ์คำว่า “TEST” ไปยัง Serial 2 บนหน้าเว็บระบบจะสั่งให้มีการถ่ายรนูปเช่นกัน
    1. พิมพ์เลขฐาน 2 ของ 0-15 ตัวอย่างเช่น 0100 ระบบจะทำการเปลี่ยนข้อมูลที่ได้รับจากฐาน 2 เป็นฐาน 16 เพื่อลดขนาดของข้อมูลที่ส่งให้เหลือ 1 byte ไปยัง serial ของ arduino 1 จะเป็นการสั่งให้กล้องหมุนไปยังมุมที่รูปนั้นอยู่ หากเลขฐานสองที่พิมพ์มาไม่มีอยู่ในหน่วยความจำที่ระบบบันทึกไว้ หรือยังไม่มีการสั่งให้ถ่ายรูปทั้งหมด (capture) ระบบจะไม่ทำตามคำสั่งนี้ 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)

รูปที่ 3.30 ข้อความที่ถูก decode แล้ว

* 1. มีการแสดงผลค่าที่ได้จากทั้ง 3 มุมหรือมุมใดมุมหนึ่งออกมาทางหลอด LED เนื่องจาก ใช้ 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

* 1. ระบบเลือกการประมวลผลภาพ มีสองแบบได้แก่ใช้ if else หรือ ML โดยการเปลี่ยน mode นั้น ทำได้โดยการ สลับสายที่เชื่อมต่ออยู่กับ Raspberry pi

รูปที่ 3.32 การสลับโหมดของการประมวลผลภาพ

* + 1. ในระบบ if else นั้นจะใช้วิธีการ ดึงค่า 4 จุดจากแต่ละ quadrant มาหา mean หากค่านั้นมากกว่า 80 แสดงว่าเป็นสีขาว แต่หากน้อยกว่า 80 แสดงว่าเป็นสีดำ (80 มาจากค่าแสดงที่วัดได้ ณ สถานที่ทดลอง) หากเลือกวิธีนี้ในการประมวลผลจะได้ทั้ง 20 ค่าสี และคำตอบว่า รูปนั้นคือรูปแบบได้
    2. ใช้ ML ประมวลผล Server 2 จะทำการส่งรูปภาพไปยัง Server ML เพื่อนำไปประมวลผลกับ model ที่ได้เตรียม ไว้แล้วตอบกลับเป็นรูปแบบของรูปนั้น หากเลือกใช้วิธีนี้ ระบบจะเรียกใช้การตรวจสอบแบบ if else เองอีกรอบหนึ่ง เพื่อให้ได้ค่าสี 20 ค่าสีด้วย

1. การติดตั้งระบบต่าง ๆ ในการทดสอบทางผู้จัดทำ ได้ทดสอบระบบ Server ต่าง ๆ ใน OS Linux (arm)
   1. หากไม่ต้องการแก้ไข config ต่าง ๆ ให้กำหนดค่าต่าง ๆ ดังนี้
      1. IP Raspberry PI1: 192.168.88.17
      2. IP Raspberry PI2: 192.168.88.19
   2. ระบบถูกพัฒนาขึ้นมาด้วยภาษา python version 3

PRETTY\_NAME="Debian GNU/Linux 10 (buster)"

NAME="Debian GNU/Linux"

VERSION\_ID="10"

* 1. ฝั่ง PC1 (Raspberry PI 1)
     1. ติดตั้ง library จาก PC1/requirements.txt ด้วยคำสั่ง

pip3 install -r requirement.txt

* + 1. แก้ไขไฟล์ PC1/config.json เป็น port ที่เชื่อมต่ออยู่กับ arduino 1

{

"port\_Arduino1": "/dev/ttyACM0",

"buadrate\_Arduino1": 115200

}

* + 1. รันไฟล์ PC1/terminal\_pc1.py ด้วยคำสั่ง

python3 terminal\_pc1/py

* 1. ฝั่ง PC 2 (Raspberry PI 2)
     1. ติดตั้ง library จาก PC2/requirements.txt ด้วยคำสั่ง

pip3 install -r requirement.txt

* + 1. 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"

}

* + 1. แก้ไขการตั้งค่าของหน้าเว็บไซต์ จากไฟล์ PC2/static/webDisplaySerial/setting.js เป็น IP ของ server ทั้ง 2 ตัว

const settings = { target:

['http://192.168.88.17:5000',

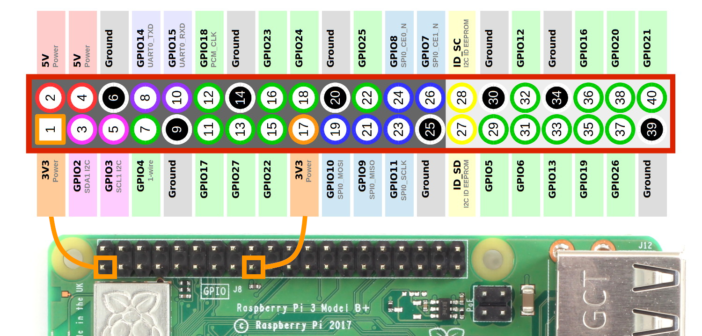
'http://192.168.88.19:5000',

'http://192.168.88.19:5000'] };

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

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

* + 1. 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

* + 1. สั่งรันไฟล์ PC2/server.py เพื่อสั่งให้ server หลักทำงาน

python3 server.py

* + 1. สั่งรันไฟล์ PC2/server\_image.py เพื่อสั่งให้ server สำหรับประมวลผล ML ทำงาน

python3 server\_image.py

1. Flowchart การทำงาน [เขียนเพิ่ม]

## 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
  + ตะกั่วบัดกรี 1 ปอนด์
  + หัวแร้งบัดกรี 1 ชิ้น
  + สายแพ 20 pin 1 เส้น
  + หัวสายแพ 20 pin 2 หัว
  + ฟิวเจอร์บอร์ด 65 x 85 cm 1 แผ่น
  + แผ่นไม้พลาสวูด 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

**ผลการดำเนินงานตามขอบเขตของโครงงาน**

จากการทดลอง มีส่วนที่สามารถทำได้ตามขอบเขตของโครงงานได้แก่

ส่วน PC1

* สามารถสั่งให้ฝั่ง PC2 เริ่มต้นการทำงาน (เก็บข้อมูลจากกล้องทุกมุมภาพ)
* แสดงผลสรุปข้อมูลรหัส Binary (0000, 0001, …, 1111) และ มุม (-45, 0, +45) ที่ได้รับจากกล้องฝั่ง PC2 มาแสดงผล
* หลังจากนั้น สามารถเลือก
  + สั่งให้ PC2 เก็บค่าข้อมูลภาพตามรหัส Binary ที่ได้
    - PC2 หมุนกล้องไปยังภาพตามรหัส Binary ที่สั่ง
    - PC2 ส่งข้อมูลจุดภาพ (ระดับสีภาพ) ตามที่กล้องหมุนไป จำนวน 20 ค่า มาแสดงผลที่ PC1
    - จำนวนข้อมูล 20 ค่า มาจาก Quadrant ละ 5 ค่า ประกอบด้วย
      * ค่าระดับสีภาพจำนวน 4 จุด พร้อมพิกัดแต่ละจุด
      * ค่าเฉลี่ยในแต่ละ Quadrant
  + สั่งให้เริ่มต้นทำงานใหม่

ส่วน PC2

* รอรับคำสั่งเริ่มการทำงานจาก PC1 (เก็บข้อมูลจากกล้องทุกมุม พร้อมวิเคราะห์ชนิดข้อมูลรหัส Binary ภาพ)
* ขั้นตอนการทำงานของ PC2 วิเคราะห์ข้อมูลชนิดข้อมูลรหัส Binary ภาพ จากกล้องในมุมต่างๆ แล้วส่งข้อมูลสรุปให้ PC1
* รอรับคำสั่งจาก PC1
  + หาก PC1 สั่งให้เก็บค่าข้อมูลภาพตามรหัส Binary
    - หมุนกล้องไปมุมภาพรหัส Binary ที่กำหนดจาก PC1
    - เก็บข้อมูลภาพ
    - ส่งข้อมูลจำนวนจุดภาพตามที่กล้องหมุนไปกลับมา PC1
  + หาก PC1 สั่งให้เริ่มต้นทำงานใหม่ จึงเริ่มทำงานใหม่ตั้งแต่ต้น

ส่วนอื่น ๆ

ขนาดกระดาษภาพ 10x10 cm จำนวน 16 แผ่น (รูปแบบรหัส Binary)

ส่วน Tx, Rx, Camera

* ใช้เป็น Arduino

ส่วนการเชื่อมต่อระหว่าง Arduino และ PC

* เป็น USB

Communication (PC)

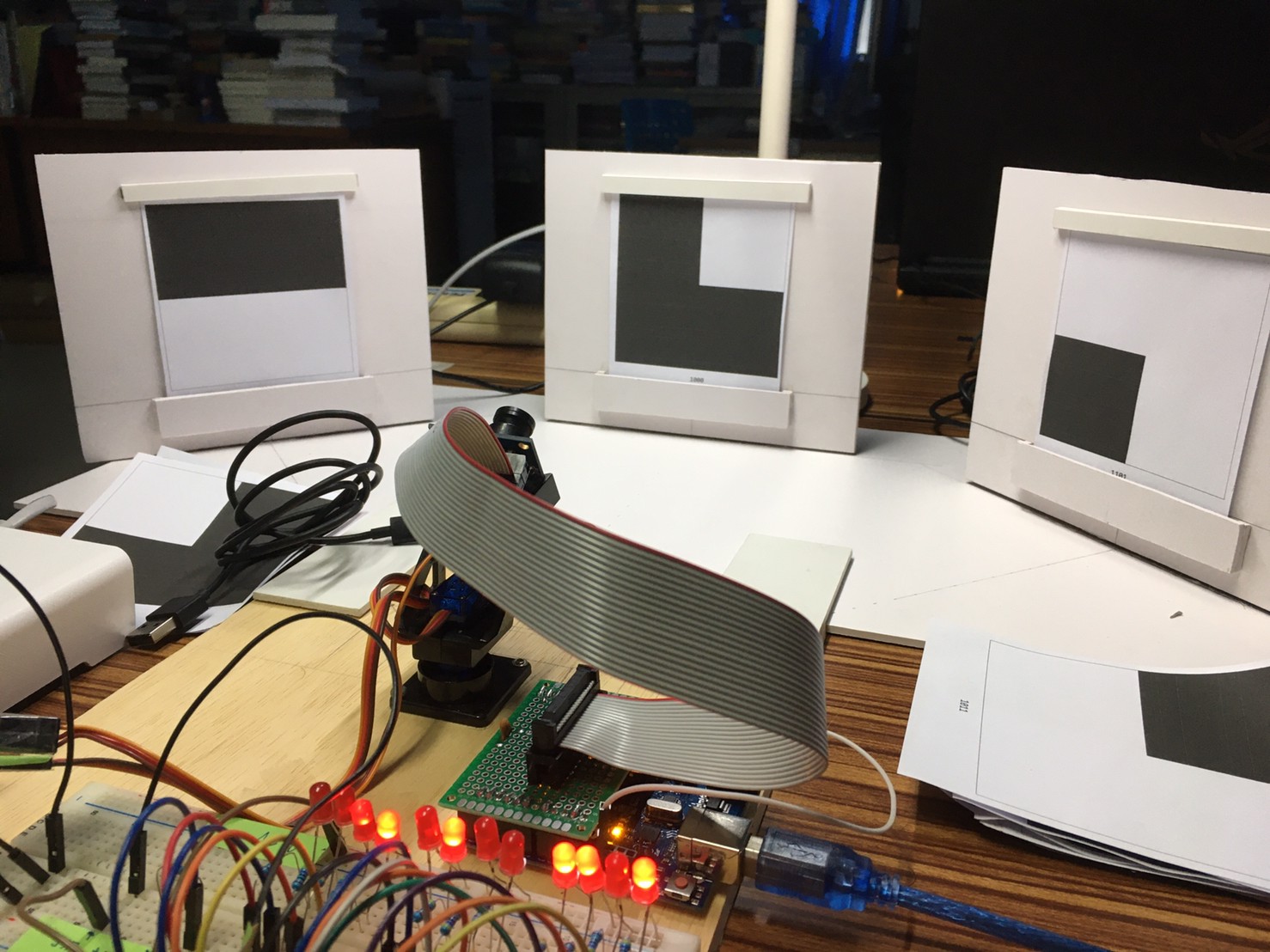
* ต้องใช้ Digital Modulation ที่เหมาะสม โดยเลือกใช้ 2-FSK
* มี Error Detection ด้วยวิธี checksum
* มีการกำหนด Frame Design ที่เหมาะสม
* มีการกำหนด Flow & Error Control โดยใช้วิธี Stop-and-Wait ARQ

ส่วนที่ไม่สามารถทำตามขอบเขตที่ตั้งไว้ได้แก่การส่งสัญญาณผ่าน Wireless ผ่านวิธี FM เนื่องด้วยสัญญาณรบกวนที่สูง ทั้งนี้ทางกลุ่มได้ทดลองหาวิธีทางแก้ปัญหาที่หลากหลายแล้วได้แก่ การทำ Band Pass Filter, การแก้ปัญหา r-slope ในซอฟต์แวร์, การคำนวณความสูงเสาสัญญาณ, การจัดตำแหน่งตัวส่งและตัวรับใหม่, เปลี่ยนตัวส่ง ตัวรับ และ Amplifier, เปลี่ยน DAC, ส่งสัญญาณในตอนกลางคืน กระนั้นแล้วทางกลุ่มก็ไม่สามารถแก้ไขปัญหาความผิดพลาดในการส่งสัญญาณผ่าน FM ในช่วงเวลาส่งงานได้ ทางกลุ่มจึงเปลี่ยนไปใช้การส่งโดยใช้สายต่อแทน 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 ค่าพิกัดจุดสีบนรูปภาพ
3. Terminal จะแสดงค่าที่ได้ที่ถูก decode แล้วเป็น 20 ค่าพร้อมกับพิกัดที่ถ่ายมา (fixed) สามารถสั่งให้กล้องหมุนไปยังองศาอื่น ๆ ต่อได้
4. สามารถสั่งงานได้จากหน้าเว็บไซต์ <http://ip_server2:5000> ด้วยเช่นกัน

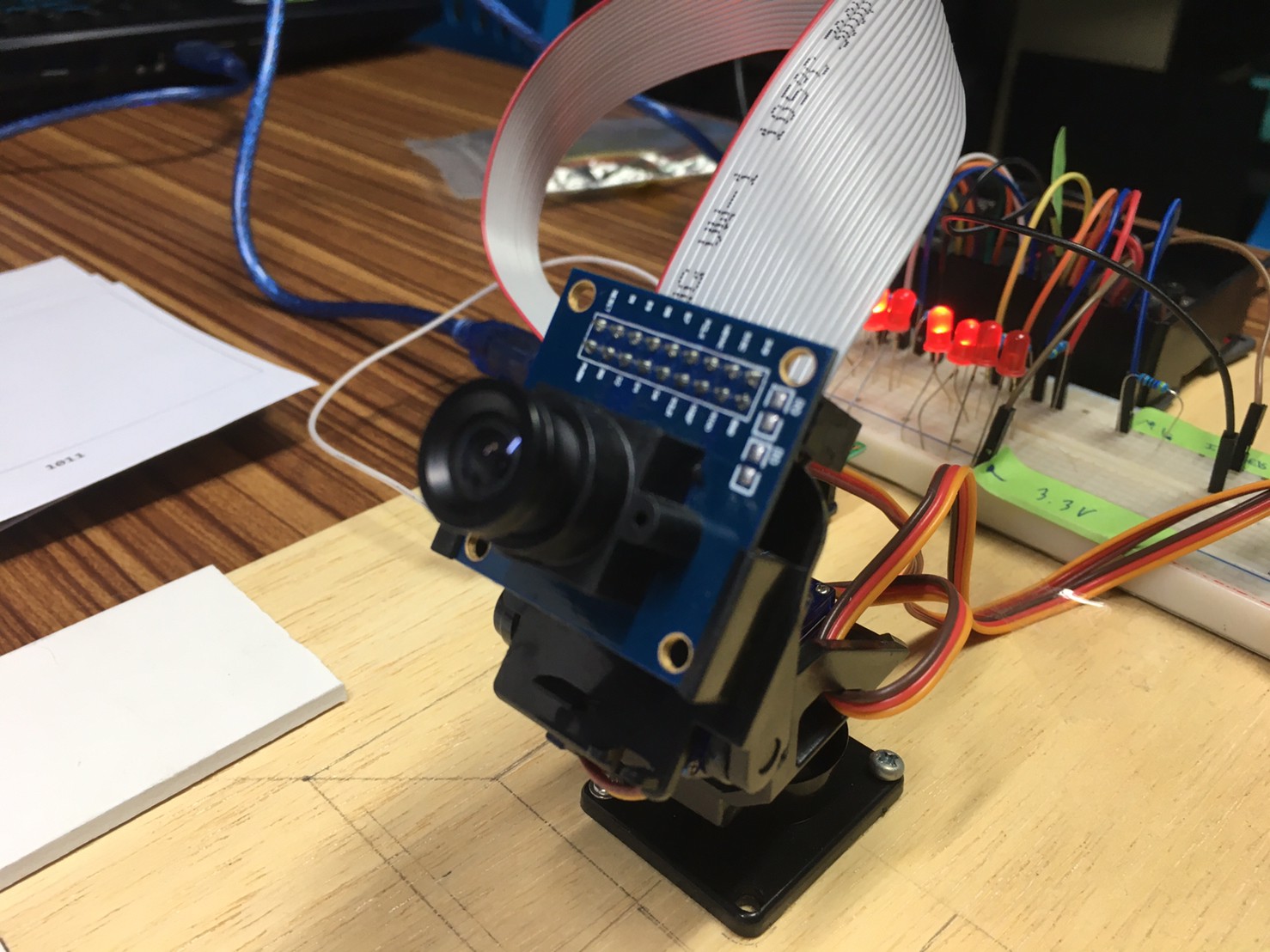
**ผลการดำเนินงาน**

****

รูปที่ 4.1 แสดงการถ่ายภาพด้วยกล้อง OV7670 กับตัวอย่างรูปแบบภาพ

****

รูปที่ 4.2 แสดงไฟ LED จากข้อมูลที่กล้องอ่านได้จากแต่ละภาพ

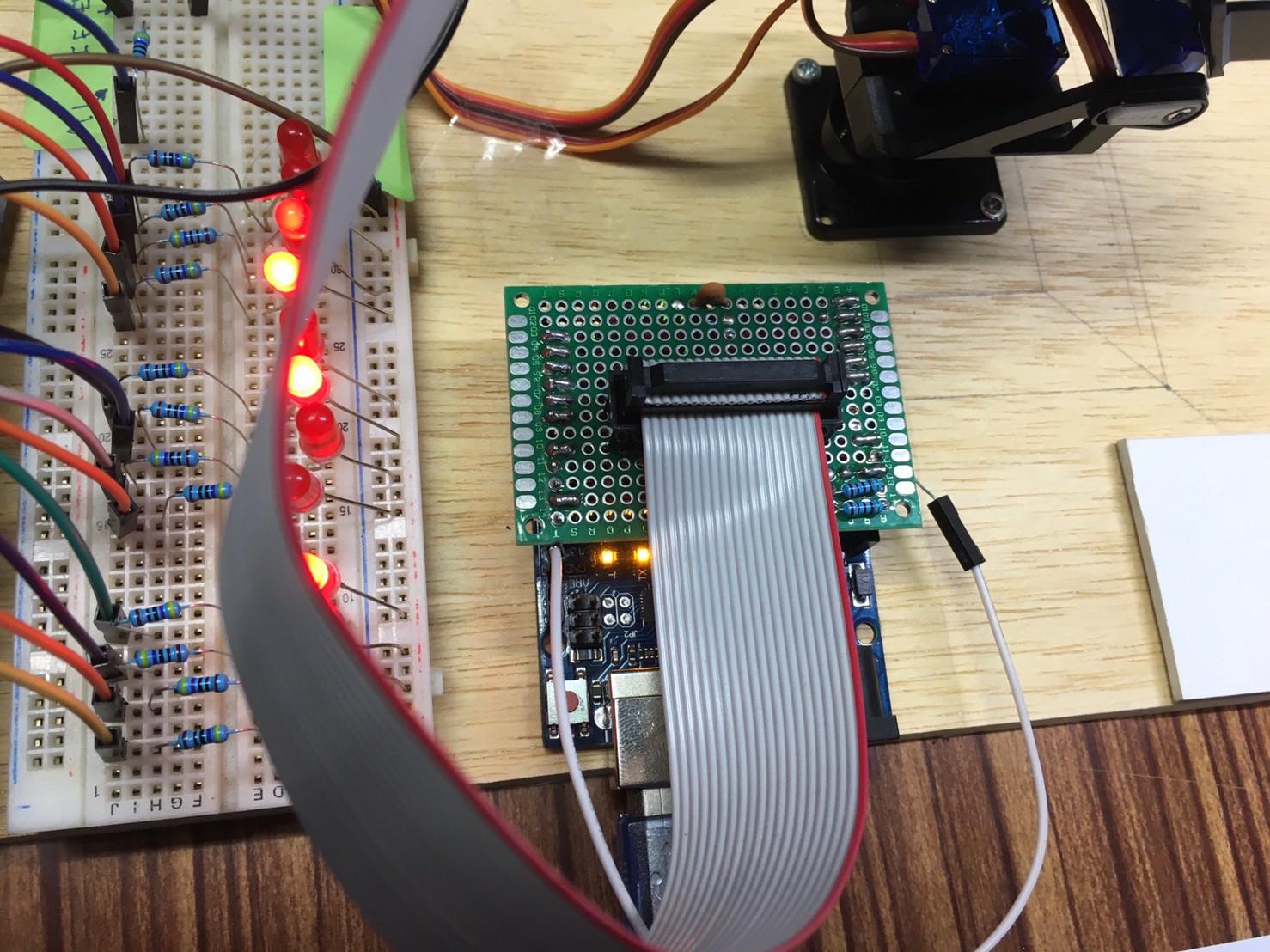
****

รูปที่ 4.3 กล้อง OV7670 Module ที่ใช้ในการถ่ายภาพ

****

รูปที่ 4.4 แสดงการทำงานระหว่าง Arduino FM & Servo และ Arduino ของกล้อง

ผ่านRaspberry Pi ตัวที่ 2

****

รูปที่ 4.5 แสดง Shield บัดกรีต่อเข้ากับ Arduino

เพื่อหลีกเลี่ยงการใช้สาย Jumper โดยไม่จำเป็น

**ภาพถ่ายที่ได้จากกล้อง OV7670 Camera Module**

* ภาพถ่ายความละเอียด 160 x 120 pixels

** **

รูปที่ 4.6 รหัสเลขฐานสอง 0000รูปที่ 4.7 รหัสเลขฐานสอง 0001

** **

รูปที่ 4.8 รหัสเลขฐานสอง 0010รูปที่ 4.9 รหัสเลขฐานสอง 0011

** **

รูปที่ 4.10 รหัสเลขฐานสอง 0100รูปที่ 4.11 รหัสเลขฐานสอง 0101

** **

รูปที่ 4.12 รหัสเลขฐานสอง 0110รูปที่ 4.13 รหัสเลขฐานสอง 0111

** **

รูปที่ 4.14 รหัสเลขฐานสอง 1000รูปที่ 4.15 รหัสเลขฐานสอง 1001

** **

รูปที่ 4.16 รหัสเลขฐานสอง 1010รูปที่ 4.17 รหัสเลขฐานสอง 1011

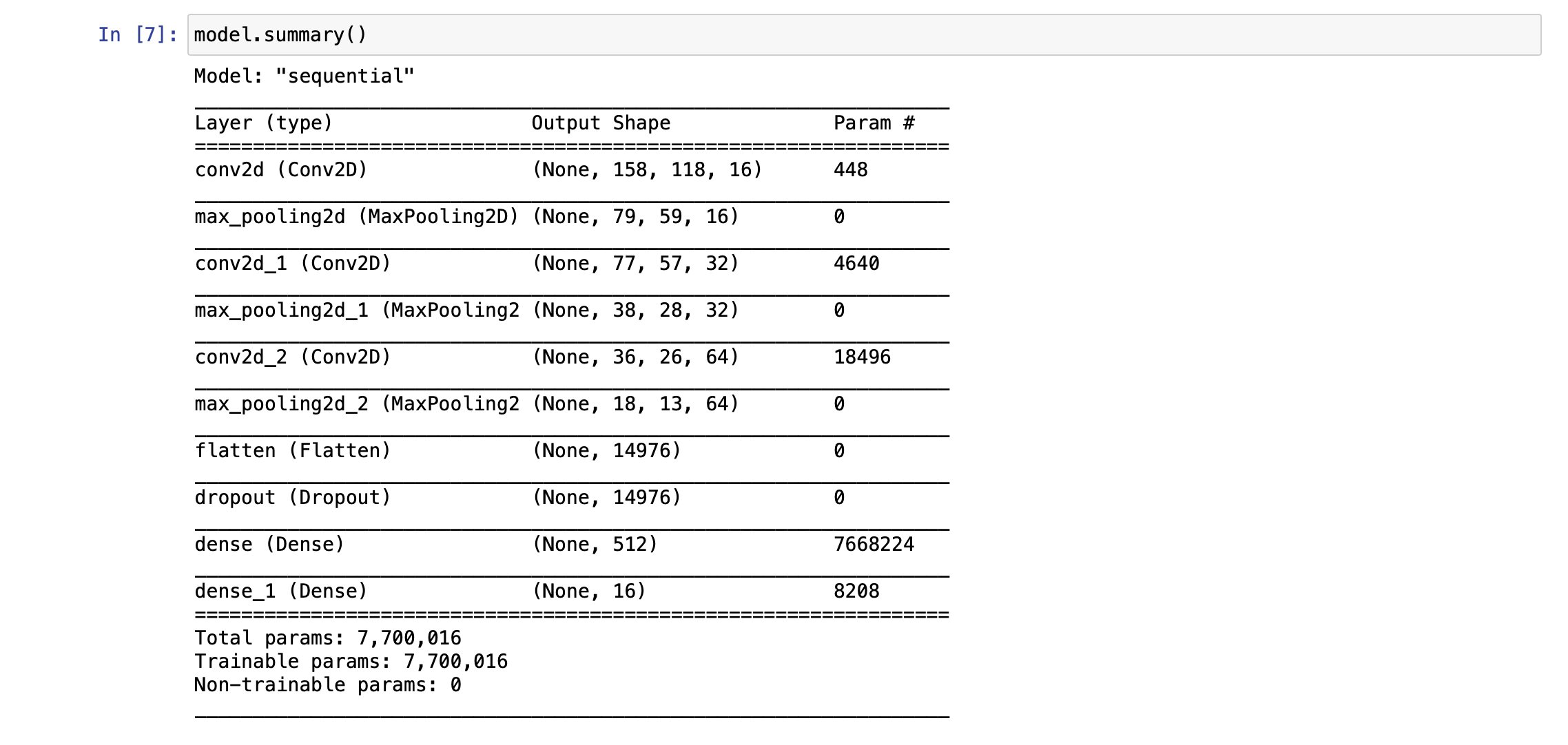
** **

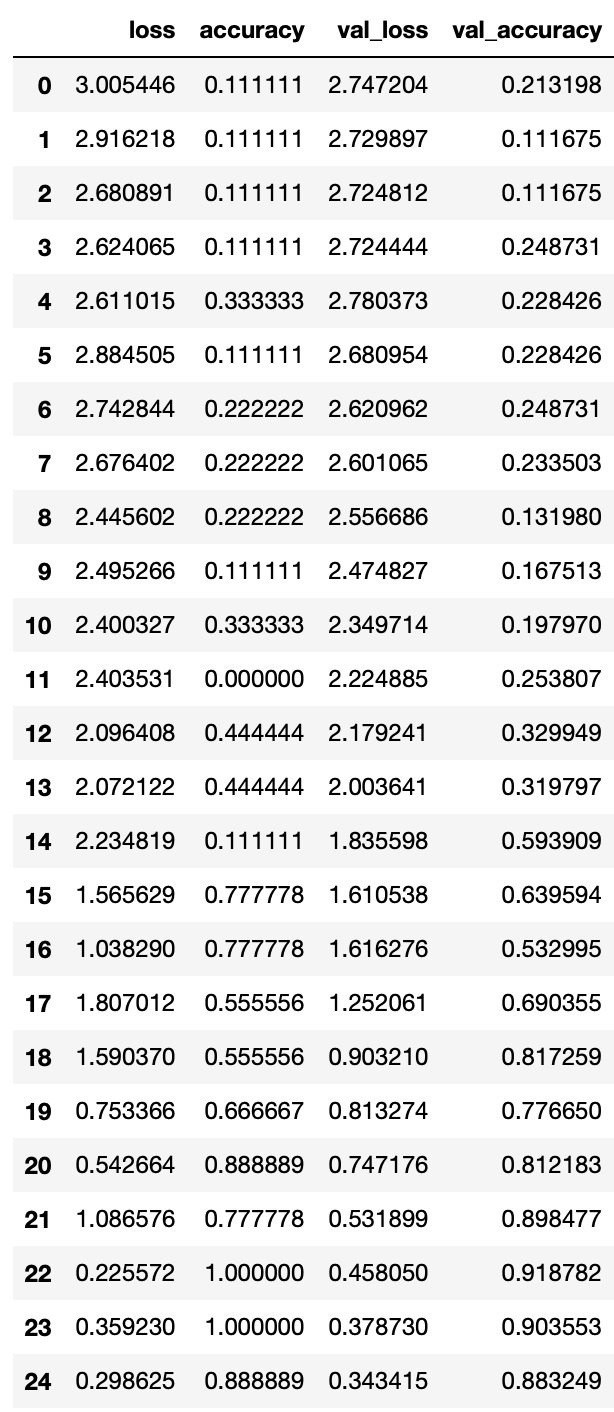
รูปที่ 4.18 รหัสเลขฐานสอง 1100รูปที่ 4.19 รหัสเลขฐานสอง 1101

** **

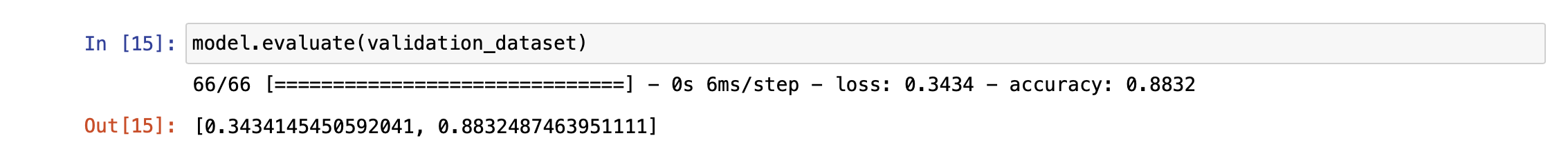
รูปที่ 4.20 รหัสเลขฐานสอง 1110รูปที่ 4.21 รหัสเลขฐานสอง 1111

## ผลลัพท์การประมวลผลภาพด้วย Machine Learning (ML)

****ตารางที่ 4.1 แสดง layer และ output ของ model

****

ตารางที่ 4.2 แสดงค่า loss, accuracy, val\_loss, val\_accuracy ของการ train แต่ละรอบ

****

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/**](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 แล้วจึงส่งข้อมูลภาพคืนให้กับ PC1 ทีละรูปโดยแปลงเลขฐาน 2 เป็นตัวอักษรแล้วค่อยแปลงกลับเป็นเลขฐาน 2 ที่ PC1 ต่อมา PC1 จะสามารถเลือกให้ถ่ายรูปใหม่โดยสั่ง “capture” หรือเลือกรูปภาพโดยการพิมพ์เลขฐาน 2 ที่ต้องการเข้าไป หากเลขฐาน 2 นั้นไม่มีอยู่ในรายการภาพที่ถ่ายได้ ก็จะไม่มีการหมุนถ่ายภาพนั้น แต่ถ้ามีอยู่ในรายการที่ถ่ายไว้ได้ในตอนแรกก็จะทำการหมุนไปยังมุมที่ภาพนั้น ๆ อยู่ ทำการถ่ายภาพนั้นใหม่แล้วส่งข้อมูลกลับออกมาเป็นตัวเลขจำนวน 20 ค่าซึ่งนำมาจากค่า 5 ค่าบนแต่ละ quadrant แล้วส่งคืนกลับมาที่ PC1 เพื่อแสดงผล นับเป็นการจบการทำงาน 1 รอบ

## ปัญหาอุปสรรคและแนวทางแก้ไข

1. สัญญาณ FM ที่รับได้มี noise มาก

แนวทางแก้ไข : ในขั้นต้นได้แก้ปัญหาโดยการทำ Band Pass Filter เพื่อตัด noise ความถี่สูง

และต่ำออกไป แต่เมื่อทำการวัดค่าสัญญาณด้วย Oscilloscope แล้วก็ยังพบว่ามี noise อยู่และไม่สามารถรับค่าที่ถูกต้องได้ กลุ่มจึงเห็นสมควรที่ทำการส่งสัญญาณผ่านสายแทน

1. การทำ Parity Bit ใช้เวลาในการทำงานมากเกินไปเพราะต้องวนทำซ้ำ 8 ครั้งต่อข้อมูล 1 ไบต์

แนวทางแก้ไข : แก้ปัญหาโดยการใช้วิธี checksum ข้อมูลและนำมา mod 2 แทนเพราะ

แทนที่ได้ด้วยข้อมูล 0 หรือ 1 และมีความสามารถในการเช็คความถูกต้องเท่ากันคือ 50%

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

แนวทางแก้ไข : แก้ปัญหาโดยการสลับเปิด-ปิดอุปกรณ์ที่ต้องสื่อสารผ่าน Serial เมื่อต้องการ

ใช้งานอุปกรณ์ใดก็เปิดใช้อุปกรณ์นั้นและปิดอุปกรณ์ที่เหลือทั้งหมด

1. การเปิด-ปิด Serial ทำให้บอร์ด Arduino รีเซ็ตตัวเอง

แนวทางแก้ไข : แก้ปัญหาโดยการต่อตัวเก็บประจุให้ขาแอโนดเข้ากับขาพิน Reset และต่อขา

แคโธดเข้ากับขา GND ของบอร์ด Arduino ทั้งนี้เมื่อต้องการอัพโหลดโคดลงบอร์ด Arduino ต้องทำการนำตัวเก็บประจุออกก่อนอัพโหลด

1. Model ในการทำ ML มีความแม่นยำต่ำ

แนวทางแก้ไข : ในขั้นต้นสามารถแก้ปัญหาได้โดยใช้การตรวจสอบ if-else แทน แต่หาก

ต้องการแก้ไขที่ model ต้องแก้ไขที่ data set โดยการเพิ่ม data set หรือการ Image Augmentation โดยมีหลายวิธีเช่น VerticalShift, Horizontal Shift, Shear, Zoom, Vertical Flip, Horizontal Flip, Rotate, Fill Mode เป็นต้น หรืออีกวิธีที่สามารถแก้ปัญหาความแม่นยำต่ำได้ก็คือ การเปลี่ยนรูปแบบ model ซึ่งมีหลายรูปแบบเช่น Logistic Regression, Naive Bayes Classifier, K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Machines เป็นต้น

1. แผ่นบัดกรีขา Pin ทั้ง 18 ที่เป็นตัวเชื่อมกลางระหว่างกล้อง 0V7670 และ Arduino เมื่อบัดกรีตามวงจรการต่อกล้องแล้วภาพไม่ชัดอย่างที่ต้องการ ภาพแตก และมี noise อยู่เยอะมาก

แนวทางแก้ไข : ในเบื้องต้นได้ลองใช้ Logic-Probe ในการวัดขาสัญญาณหลังจากสัญญาณ

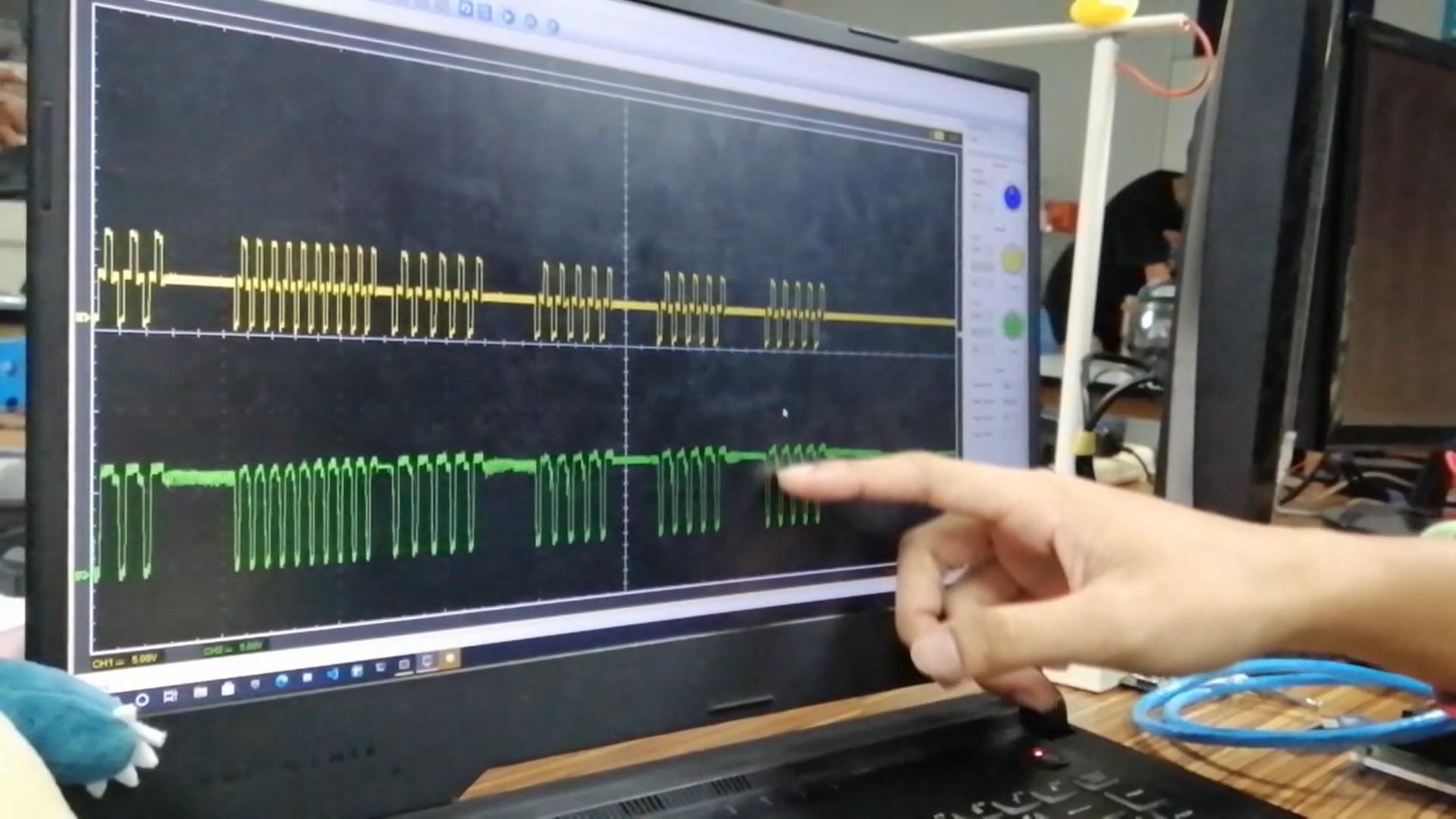
ที่ผ่านบอร์ดบัดกรี จึงพบว่า ขา V-Sync มีปัญหา เนื่องจากขาสัญญาณนี้ ปกติจะเป็น Low และเป็น High เมื่อข้อมูลถูกส่งเรียบร้อยแล้ว ซึ่งข้อมูลนี้วิ่งด้วยความเร็วกว่า 8 ล้านครั้งต่อวินาที (อิงจากโค้ดการส่งข้อมูลจากกล้องถึง Arduino) และใช้สายไฟบัดกรีธรรมดา จึงทำให้เกิด Crosstalk ไปรบกวนสัญญาณของเส้น V-Sync ทำให้ภาพที่ได้นั้น 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 ควรลดทอนมาเป็นการส่งสายแทน เพื่อทำให้ยังคงสภาพการทำงานทั้งหมดให้ยังสามารถดำเนินต่อได้อย่างปกติ เมื่อส่งเสร็จ ทางผู้จัดทำได้ทำการเร่งแก้ไขในส่วนต่างๆ เช่นปรับอุปกรณ์ให้ดีขึ้น กำหนด band pass filter ให้ดีขึ้น และใช้ช่วงเวลาที่ไม่มีคลื่นรบกวนอยู่มากเป็นช่วงเวลากลางคืน พบว่าสามารถส่ง FM ได้เป็นอย่างดี

1. สามารถส่งผ่านคลื่น FM ได้แทนการใช้สาย
   1. อธิบายหลักการทำงาน FM



รูปที่ 6.1 สามารถดูในคลิป VDOdemo\_Group1.mp4 ได้ที่วินาทีที่ 7:55 - 10:55 นาที

* 1. 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 จักกริช พฤษการ. พิมพ์ครั้งที่ 1. กรุงเทพฯ : สำนักพิมพ์ท็อป.

Jirasak Sittigorn. (2563). **Data Communication Assignment 2563**. [ออนไลน์].

เข้าถึงได้จาก : <https://docs.google.com/document/d/1lCAKcCvDOUd78t_>

YZG8W5tjDkQTTrvopiaaOMNXXE0I/edit (วันที่ค้นข้อมูล : 5 พฤศจิกายน 2563).

Jirasak Sittigorn. (2563).  **ขั้นตอนการใช้งาน OV7670 (โมดูลกล้อง)**. [ออนไลน์].

เข้าถึงได้จาก : [https://docs.google.com/document/d/1kdxclTAXJckRb7](https://docs.google.com/document/d/1kdxclTAXJckRb7b)b

2CWud6TY4N25vYgkQhOkC9wr5MgM/edit. (วันที่ค้นข้อมูล : 5 พฤศจิกายน 2563).

**Opening serial ports**. (2563). [ออนไลน์]. เข้าถึงได้จาก :

[https://pyserial.readthedocs.io/en/latest/shortintro.htm](https://pyserial.readthedocs.io/en/latest/shortintro.html)l.

(วันที่ค้นข้อมูล : 7 พฤศจิกายน 2563).

**Getting Started with Images**. (2563). [ออนไลน์]. เข้าถึงได้จาก : [https://opencv-python-](https://opencv-python-tutroals)

[tutroals](https://opencv-python-tutroals).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-](https://axotron.se/blog/crosstalk-pro)

[pro](https://axotron.se/blog/crosstalk-pro)blems-when-running-i2c-signals-in-a-cable.

(วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

**การแทรกสัญญาณข้าม**. (2563). [ออนไลน์]. เข้าถึงได้จาก : th.wikipedia.org/wiki/การแทรก

สัญญาณข้าม. (วันที่ค้นข้อมูล : 9 พฤศจิกายน 2563).

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

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

**ในโทรคมนาคม Crosstalk คืออะไร?**. (2563). [ออนไลน์]. เข้าถึงได้จาก : [https://www.netin](https://www.netinbag.com/)

[bag.com/](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://towards](https://towardsdatascience.com/applied-deep-learning-part-4-convolutonal-neural-networks-584bc134c1e2)

[datascience.com/applied-deep-learning-part-4-convolutonal-neural-networks-584bc134c1e2](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-Under>

standing-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/e0b8a3e0b8b9e0b89be0b981e0b89ae0b89

ae0b8a3e0b8b2e0b8a2e0b887e0b8b2e0b899.pdf.(วันที่ค้นข้อมูล : 23 พฤศจิกายน 2563).

# ภาคผนวก

**โค้ดทีใช้ในการทำงาน**

**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 0x0a /\* Product ID MSB \*/

#define REG\_VER 0x0b /\* Product ID LSB \*/

#define REG\_COM3 0x0c /\* 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 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 /\* Clocl 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 in0x58.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 0x0a /\* Product ID MSB \*/

#define REG\_VER 0x0b /\* Product ID LSB \*/

#define REG\_COM3 0x0c /\* 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 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 /\* Clocl 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 downsamp/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, 0x0a },

{ 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 rsvd 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, 0x0a },

/\* 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) {

arduinoUnoInut();

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\_recever\_arduino = []

send\_msg\_Arduono = ""

def serial\_run():

global stop\_thread

global string\_recever\_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")

postition = [[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(postition[idy][0] + (idx \* 4)) + ',' +

str(postition[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\_recever\_arduino

for idx in range(len(string\_recever\_arduino)):

if len(string\_recever\_arduino):

yield "data:" + string\_recever\_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\_recever\_arduino2 = []

send\_msg\_Arduono3 = ""

send\_msg\_Arduono2 = ""

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\_Arduino2.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\_Arduino2.inWaiting():

#print("Have data!")

serialString = serial\_Arduino2.readline()

print(

colored(("Data in serial", serialString),

"magenta"))

serial\_Arduino2.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\_recever\_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\_Arduono2 = command

return 'OK'

#arduino 2

@app.route('/stream2')

def stream():

def read\_process():

while True:

global string\_recever\_arduino2

for idx in range(len(string\_recever\_arduino2)):

if len(string\_recever\_arduino2):

yield "data:" + string\_recever\_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\_Arduono2

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)