การพัฒนาการจำแนกและจดจำวัตถุใต้น้ำโดยวิธีการประมวลผลภาพดิจิตอล

Development of Object analyze and Recognition by Digital Image Processing

นายเจตน์จักร เนตรสุวรรณ รหัส 54360612
นายสุทธิรงค์ อยู่เอม รหัส 54360902
นายจิณณวัตร เรืองนิ่ม รหัส 54363156

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมเครื่องกล ภาควิชาวิศวกรรมเครื่องกล
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2557

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน การพัฒนาการจำแนกและจดจำวัตถุใต้น้ำโดยวิธีการประมวลผลภาพดิจิตอล

ผู้ดำเนินโครงงาน นายเจตน์จักร เนตรสุวรรณ รหัส 54360612
นายสุทธิรงค์ อยู่เอม รหัส 54360902
นายจิณณวัตร เรือนิ่ม รหัส 54363156

ที่ปรึกษาโครงงาน อาจารย์ชูพงศ์ ช่วยเพ็ญ

สาขาวิชา วิศวกรรมเครื่องกล

ภาควิชา วิศวกรรมเครื่องกล

ปีการศึกษา 2557

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมเครื่องกล

คณะกรรมการสอบโครงงานวิศวกรรมเครื่องกล

.....ที่ปรึกษา
(อาจารย์ชูพงศ์ ช่วยเพ็ญ)

.....กรรมการ
(ผศ.ดร.อนันต์ชัย อยู่แก้ว)

.....กรรมการ
(อาจารย์สุรเจษฎ์ สุขไชยพร)

ชื่อหัวข้อโครงงาน	การพัฒนาการจำแนกและจดจำวัตถุใต้น้ำโดยวิธีการประมวลผลภาพดิจิตอล			
ผู้ดำเนินโรงงาน	นายเจดน์จักร	เนตรสุวรรณ	รหัส	54360612
	นายสุทธิรงค์	อยู่เอม	รหัส	54360902
	นายจิณณวัตร	เรือนิ่ม	รหัส	54363156
ที่ปรึกษาโครงงาน	อาจารย์ชูพงศ์	ช่วยเพ็ญ		
สาขาวิชา	วิศวกรรมเครื่องกล			
ภาควิชา	เครื่องกล			
ปีการศึกษา	2557			

บทคัดย่อ

โครงงานมีวัตถุประสงค์เพื่อพัฒนาการจำแนกและจดจำวัตถุใต้น้ำโดยวิธีการประมวลผลภาพดิจิตอลเพื่อเป็นระบบนำร่องให้กับยานสำรวจใต้น้ำ สำหรับภารกิจใต้น้ำที่มนุษย์ไม่สามารถเข้าถึงได้ โดยได้ออกแบบให้สามารถจำแนกและจดจำตำแหน่ง และสีของวัตถุที่อยู่ใต้น้ำได้

การดำเนินงานพัฒนาเริ่มจากการศึกษาเกี่ยวกับอุปกรณ์และความรู้พื้นฐานเกี่ยวกับการประมวลผลภาพดิจิตอล เช่น Raspberry Pi ซึ่งเป็นฮาร์ดแวร์หลักที่ใช้ในการการเขียนโปรแกรม และอนุกรมสีของภาพ เช่น อนุกรมสีแบบ RGB ซึ่งเป็นอนุกรมสีที่ใช้ระบุชนิดของสี และอนุกรมสีแบบ HSV ซึ่งเป็นอนุกรมสีที่สามารถบอกชนิดและความเข้มของสีได้ศึกษาซอฟแวร์ที่ใช้ในการวิเคราะห์ข้อมูลที่ได้รับมาจากกล้องซึ่งเชื่อมต่ออยู่กับตัว Raspberry Pi ยกตัวอย่างเช่น OpenCV, C++, Wiring Pi เป็นต้น ในการประมวลผลภาพ ที่ต้องการแบ่งภาพดิจิตอลออกเป็นส่วนๆ (Image segmentation) โดยวิธีการ ใช้ Thresholds ซึ่งจะแบ่งภาพเป็น binary(ขาว – ดำ) และ ยังสามารถใช้ binary image ในการเติมเติม รูปภาพดิจิตอล ที่ไม่สมบูรณ์หรือ ส่วนเกินของภาพ ด้วยวิธีการ Erosion(การกัดเซาะ) และ Dilation(การขยาย) และยังสามารถ สร้าง line contour เพื่อเพิ่มกลุ่มพื้นที่ ที่ต้องการความเด่นชัด

Project title	Development of Object analyze and Recognition by Digital Image Processing			
Author	Mr. Jadjak	Netsuwan	ID	54360612
	Mr. Sutthirong	Yoo-em	ID	54360902
	Mr. Jinnawat	Ruangnim	ID	54363156
Project advisor	Mr. Choopong Chuaypen			
Major	Mechanical Engineering			
Department	Mechanical Engineering			
Academic year	2014			

Abstract

This project to aim to developing the identification and recognition the underwater object by using digital image processing for the navigation system of the underwater drone to work in the mission that people can't do by themselves. The design of the drone must make the drone can detect and remember the location and the object color in the water.

The development is started with learn about equipment and basic knowledge about digital image processing. For example, "The Raspberry Pi" that is the main hardware for the program writhing and the color series likes "The RGB color series" that use to identify the type of color and "The HSV color series" that can identify the type and the saturation of the color. And learn about the software that used to analysis the information that came from the camera that connect to the Pi. For example, OpenCV, C++, and WiringPi etc. To processing the image that wanted to slice image into pieces (image segmentation) by using thresholds that make the image into binary (black-white) and we can use this binary image to filled the digital image that incomplete or the image

กิตติกรรมประกาศ

คณะผู้จัดทำขอขอบพระคุณผู้ที่ให้การสนับสนุนข้อมูลในด้านต่างๆ จนทำให้การศึกษาโครงงานนี้สำเร็จลุล่วงไปด้วยดี ดังจะกล่าวต่อไปนี้

อาจารย์ชูพงศ์ ช่วยเพ็ญ อาจารย์ที่ปรึกษาโครงงาน อาจารย์ประจำภาควิศวกรรมเครื่องกล มหาวิทยาลัยนเรศวร ที่ให้คำปรึกษา แนะนำ และให้ข้อมูลที่เป็นประโยชน์เกี่ยวกับวิธีประมวลผลภาพดิจิตอลสำหรับการจำแนกและจดจำวัตถุ และเอื้อเฟื้อห้องทำงานสำหรับการพัฒนาโครงงานนี้

คณะวิศวกรรมศาสตร์และอาจารย์ทุกท่านในคณะวิศวกรรมศาสตร์ ที่ได้ประสิทธิ์ประสาทความรู้แก่คณะผู้ดำเนินโรงการตลอดจนให้คำแนะนำปรึกษาในเรื่องต่างๆ

สุดท้ายนี้ คณะผู้ดำเนินงานหวังว่าโครงงานวิศวกรรมเครื่องกลฉบับนี้ จะเป็นประโยชน์แก่ผู้ที่ศึกษาเรื่องการพัฒนาการจำแนกและจดจำวัตถุใต้น้ำโดยวิธีการประมวลผลภาพดิจิตอล และถ้าเกิดข้อผิดพลาดประการใดจากการดำเนินงานโครงงานวิศวกรรมเครื่องกลฉบับนี้ คณะผู้ดำเนินงานต้องกราบขออภัยมา ณ ที่นี้ด้วย

คณะผู้ดำเนินโครงงาน
นายเจตน์จักร เนตรสุวรรณ
นายสุทธิรงค์ อยู่เอม
นายจิณณวัตร เรืองนิ่ม
เมษายน2558

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

สารบัญสัญลักษณ์และอักษรย่อ

mp = Megapixel

RGB = Red green blue

HIS = Hue Saturation Intensity

BCM = Broadcom

SoC = System on a Chip

MHz= Mega Hertz

RAM = Random Access Memory

GPU = Graphic Processor Unit

MB = Megabyte

USB = Universal Serial Bus

PAL = Phase Alternating Line

NTSC = National Television System Committee

DSI = Display Serial Interface

LCD = Liquid Crystal Display

HDMI = High-Definition Multimedia Interface

SD = Secure Digital

MMC = Multimedia Card

SDIO = Secure Digital Output

GPIO = General purpose input/output

UART = Universal asynchronous receiver/transmitter

V =Volt

SPI = Serial Peripheral Interface

mm = Millimeter

GUI = Graphical User Interface

UTP = Unshield Twisted Pairs

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

โครงการนี้เป็นส่วนหนึ่งของโครงการพัฒนาหุ่นยนต์สำรวจใต้น้ำ ซึ่งจะมุ่งเน้นไปที่การประมวลผลภาพเพื่อใช้ในการควบคุมตำแหน่งและการทำงานตามคำสั่งแบบอัตโนมัติ ทั้งนี้เนื่องด้วยการถ่ายทอดข้อมูล ทั้งข้อมูลภาพและเสียง ข้อมูลคำสั่ง ตลอดจนข้อมูลอื่น ระหว่างหุ่นยนต์สำรวจใต้น้ำกับผู้ใช้งานหรือผู้ควบคุม หากไม่ใช้สายส่งสัญญาณจะทำได้ยาก ทั้งเนื่องจากการรบกวนจากสภาพแวดล้อม การสูญเสียสัญญาณข้อมูล ความล่าช้าของข้อมูลที่ได้รับ ซึ่งจะทำให้สั่งการควบคุมหุ่นยนต์สำรวจใต้น้ำแบบไร้สายเป็นไปได้ยาก หรืออาจเป็นไปไม่ได้เลย

การระบุตำแหน่งของหุ่นยนต์สำรวจใต้น้ำ การระบุตำแหน่งของสิ่งกีดขวาง หรือการระบุตำแหน่งของวัตถุเป้าหมายสามารถทำได้โดยใช้เซนเซอร์ได้ชนิดต่างๆ แต่จะทำได้เฉพาะเจาะจงเป็นอย่างๆ ตามแต่ชนิดของเซนเซอร์นั้นๆ ข้อมูลภาพซึ่งถือว่าเป็นค่าจากเซนเซอร์แสงแบบหนึ่งหรือก็คือกล้องถ่ายภาพหรือกล้องวีดิโอมีข้อดีที่เหนือกว่าข้อมูลจากเซนเซอร์อื่นๆก็คือ เราสามารถทำการประมวลภาพเพื่อให้ได้ข้อมูลอื่นๆได้หลากหลาย เช่น การจำแนกชนิด การจำแนกสี การจำแนกขนาด การระบุระยะห่างของวัตถุได้ เป็นต้น ซึ่งจะเห็นได้ว่าเพียงกล้องตัวเดียวเราสามารถวิเคราะห์ค่าการวัดต่างๆ ได้หลากหลาย ทั้งนี้เทคนิคการประมวลผลและวิเคราะห์ภาพถ่ายมีหลักการและกระบวนการต่างๆมากมาย เช่นการวิเคราะห์และรู้จำสี การวิเคราะห์และรู้จำชนิดของวัตถุ การระบุตำแหน่งของวัตถุในสามมิติ กระปรับปรุงคุณภาพของข้อมูลภาพ

ดังนั้นในหัวหัวข้อโครงการนี้จึงมุ่งเน้นเพื่อศึกษาการวิเคราะห์และประมวลผลข้อมูลภาพสำหรับหุ่นยนต์สำรวจใต้น้ำ ในสภาพแวดล้อมที่กำหนด เพื่อให้ได้มาซึ่งข้อมูลที่สามารถนำไปใช้ในการควบคุมการเคลื่อนที่และการทำงานตามคำสั่งแบบอัตโนมัติ

1.2 วัตถุประสงค์ของการศึกษา

1.2.1 เพื่อศึกษาทฤษฎี หลักการ และกระบวนการต่างๆ ที่ใช้ในการวิเคราะห์และประมวลผลภาพ เพื่อใช้ในการวิเคราะห์ รู้จำ และจำแนกวัตถุต่างๆได้

1.2.2 เพื่อศึกษาทฤษฎี หลักการ และกระบวนการต่างๆ ที่ใช้ในการวิเคราะห์และประมวลผลภาพ เพื่อใช้ในการวิเคราะห์ตำแหน่งของวัตถุต่างๆได้

1.2.3 เพื่อศึกษาและประยุกต์ใช้ทฤษฎี หลักการ และกระบวนการต่างๆ ข้างต้นกับข้อมูลภาพใต้น้ำในสภาพแวดล้อมที่กำหนด และค่าความสำเร็จเมื่ออยู่ภายใต้สภาพแวดล้อมอื่นที่สร้างเพื่อทดสอบ

1.2.4 สร้างกระบวนการและปรับปรุงกระบวนการประมวลผลภาพที่เหมาะสมกับระบบของหุ่นยนต์สำรวจใต้น้ำที่ออกแบบ

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ทฤษฎี หลักการ และกระบวนการต่างๆ ที่ใช้ในการวิเคราะห์และประมวลผลภาพ เพื่อใช้ในการวิเคราะห์ รู้จำ และจำแนกวัตถุต่างๆได้ และเพื่อใช้ในการวิเคราะห์ตำแหน่งของวัตถุต่างๆได้

1.3.2การพัฒนาและการประยุกต์ใช้ ทฤษฎี หลักการ และกระบวนการต่างๆ ที่ใช้ในการวิเคราะห์และประมวลผลภาพ เพื่อใช้ในการวิเคราะห์ รู้จำ และจำแนกวัตถุต่างๆ และเพื่อใช้ในการวิเคราะห์ตำแหน่งของวัตถุต่างๆ ใต้น้ำได้

1.3.3 เรียนรู้ พัฒนา และประยุกต์กระบวนการในการประมาณผลภาพบนอุปกรณ์แบบฝังตัว (Embedded system)

1.4 ขอบเขตของโครงงาน

1.4.1 พัฒนากระบวนการเพื่อใช้ในการวิเคราะห์และประมวลผลภาพ เพื่อการวิเคราะห์ รู้จำ และจำแนกวัตถุต่างๆ ใต้น้ำ

1.4.2 พัฒนากระบวนการเพื่อใช้ในการวิเคราะห์และประมวลผลภาพ เพื่อการวิเคราะห์ตำแหน่งของวัตถุต่างๆ ใต้น้ำ

1.4.3 ออกแบบรูปแบบข้อมูลที่จะใช้ในการสื่อสารเพื่อส่งข้อมูลที่ประมวลผลได้ให้กับอุปกรณ์อื่นๆ ในระบบฝังตัว

1.5 แผนการดำเนินงาน

ตารางแสดงขั้นตอนการดำเนินงาน

กิจกรรม	2557					2558				
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1.ศึกษาทฤษฎีรวมถึงข้อมูลรายละเอียดต่างๆของงานวิจัยที่เกี่ยวข้อง										
2 สร้างโปรแกรมที่ใช้วิเคราะห์ภาพ										
3 ทำการทดลองกับภาพนิ่งและภาพเคลื่อนไหวกับโปรแกรมที่สร้างขึ้น										
4 เปรียบเทียบผลที่ได้จากโปรแกรมกับสิ่งที่เห็น										
6. วิเคราะห์ผลข้อมูล										
7. สรุปผลและจัดทำรายงาน										

1.6 งบประมาณ

1.6.1 ค่ากระดาษ 500 บาท

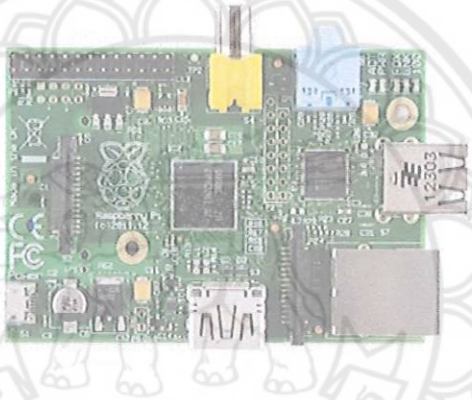
1.6.2 ค่าจัดทำรูปเล่ม 500 บาท

1.6.3 ค่าวัสดุอุปกรณ์อื่นๆ 2000 บาท

บทที่ 2

หลักการและทฤษฎีเบื้องต้น

2.1 คุณสมบัติต่างๆของอุปกรณ์

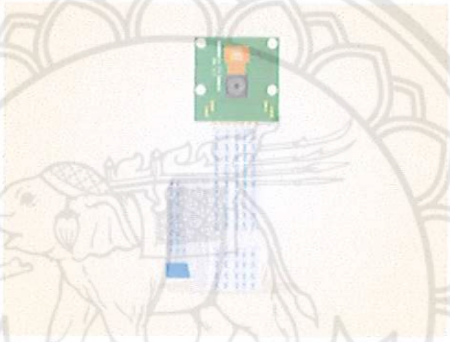


รูปที่ 2.1 Raspberry Pi
ที่มา http://www.thaieasyelec.com/products/
development-boards/raspberry-pi-512mb-model-b-detail.html

2.1.1 Raspberry Pi รุ่น Raspberry Pi 512 MB Model B Dimensions: 85.60mm x 56mm x21mm ภายในประกอบด้วย

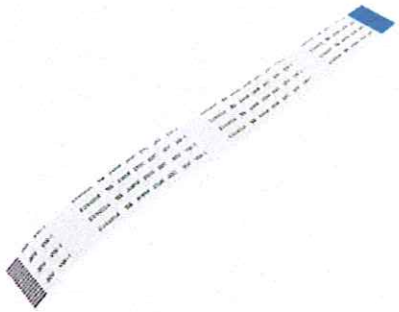
- Broadcom BCM2835 SoC
- 700 MHz ARM1176JZF-S core CPU
- Broadcom VideoCore IV GPU
- 512 MB RAM
- 2 x USB2.0 Ports with up to 1.2A output
- Video Out via Composite (PAL and NTSC), HDMI or Raw LCD (DSI)
- Audio Out via 3.5mm Jack or Audio over HDMI
- 10/100 Ethernet (RJ45)
- Storage: SD/MMC/SDIO

- Low-Level Peripherals: Power Requirement: 5V @ 700 mA via MicroUSB or GPIO Header ประกอบด้วย
  - 8 x GPIO
  - UART
  - I2C bus
  - SPI bus with two chip selects
  - +3.3V
  - +5V
  - Ground



รูปที่ 2.2 โมดูลของกล้อง Raspberry Pi
ที่มา https://www.raspberrypi.org/products/camera-module/

2.1.2 Raspberry PI camera module Element14 รุ่น RPI CAMERA BOARD Dimension: 25mm x20mm 9mm ประกอบด้วย
- กล้องขนาดความละเอียด 5 ล้าน pixel



รูปที่ 2.3 สาย flat cable 15 Way
ที่มา http://www.modmypi.com/raspberry-pi/camera/rpberry-pi-camera-board-replacement-cable-150mm

2.1.3 15-Way 150mm flat cable สายสำหรับต่อกล้องกับบอร์ด CSI

รูปที่ 2.4 UTP cable

ที่มา http://www.icidu.com/media/productdownloads/N-707538/images/N-707538%20UTP%20CAT%205e%20Cable%20RJ45%20M%20RJ45%20M%201m%20L.jpg

2.1.4 UTP Cable

2.1.5 แหล่งจ่ายไฟฟ้ากระแสตรง เข้าบอร์ด ขนาด 5V 700mA

รูปที่ 2.5 HDMI to VGA converter

ที่มา http://titan-ice.co.za/hdmi-to-vga-d-sub-display-adapter-converter.html

2.1.6 HDMI to VGA converter ใช้แปลงสัญญาณภาพจาก HDMI เข้าสู่ VGA เพื่อง่ายต่อการต่อเข้ากับจอคอมพิวเตอร์ทั่วไป

2.1.7 จอคอมพิวเตอร์ที่รับสัญญาณจาก AVI หรือ HDMI ได้

รูปที่ 2.6 SDHC CARD

ที่มา https://www.jib.co.th/web/index.php/product/readProduct/6969/20/16-GB-SD-CARD-KINGSTON-CLASS-10

2.1.8 SDHC Kingston 16 GB class10 ใช้สำหรับบันทึกข้อมูลที่มีปริมาณมากของวิดีโอและรูปภาพที่ได้จากการประมวลผล และเหตุผลที่เลือกใช้ class10 เนื่องมาจากต้องการความเร็วในการถ่ายโอนข้อมูลระหว่างการ์ด

2.2 หลักการทำงานของอุปกรณ์ต่างๆ

2.2.1 Raspberry pi การทำงานหลักๆ ที่สามารถใช้ได้อย่างมีประสิทธิภาพ อาทิเช่น

- มี GPIO ทำให้สามารถเชื่อมต่อกับ sensor หรือส่งค่าออกไปภายนอกได้แบบ Microcontroller
- มี Protocol SPI UART I2C ให้ใช้
- สามารถทำงานเป็น Server ได้

2.2.2 โมดูลกล้อง ใช้ในการรับภาพและส่งไปให้หน่วยประมวลผล

2.3 ระบบปฏิบัติการที่เกี่ยวข้อง

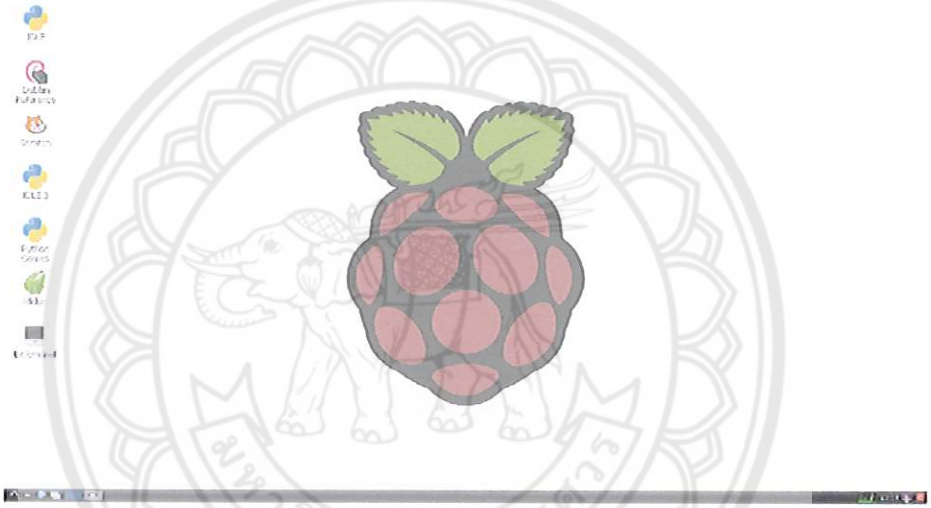
2.3.1 Ubuntu Linux ใช้ในการปรับปรุงแก้ไขในส่วนซอร์ฟแวร์ภายใน Raspberry Pi

2.3.2 Microsoft window 8 ใช้ในการรันโปรแกรม visual machine เพื่อรันระบบปฏิบัติการ Ubuntu อีกที

2.3.3 RASPBIAN ใช้เป็นระบบปฏิบัติการรันบนบอร์ด raspberry เพื่อใช้ในการประมวลผลต่างๆ

2.4 หลักการทำงานของบอร์ด Raspberry Pi

Raspberry pi สามารถทำงานเหมือนคอมพิวเตอร์ตัวหนึ่ง ซึ่งสามารถต่อ TV keyboard หรือ mouse ได้ด้วย โดยใช้ระบบปฏิบัติการ open source ได้หลายแบบ และสามารถรันแบบ terminal โดยไม่ต้องเข้าระบบปฏิบัติการเพื่อความรวดเร็วในการสั่งงาน ในที่นี้จะใช้ระบบปฏิบัติการ RASPBIAN

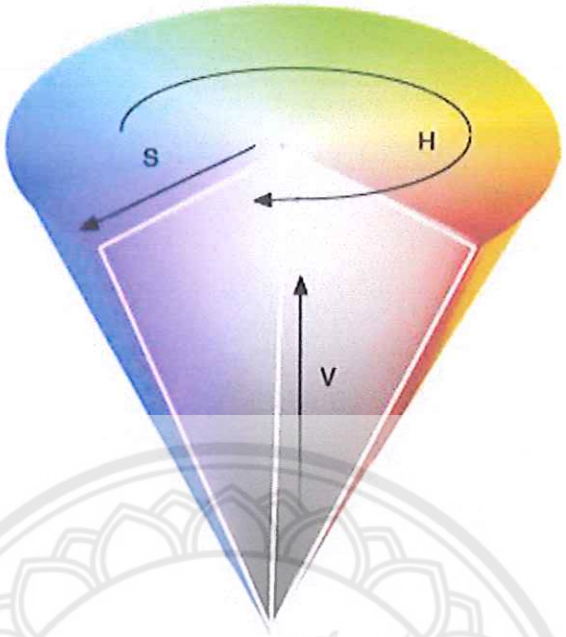


รูปที่ 2.7 interface ของ RASBIAN

ที่มา http://www.blaess.fr/christophe/wp-content/uploads/2012/08/screen-01.png

2.5 โมเดลสี HSV (Hue, Saturation และ Value)

เป็นโมเดลสีที่ใช้ค่า HSV (Hue, Saturation และ Value) ในการกำหนดค่าต่างๆ และการเปรียบเทียบสีโดยรูปทรงกรวย ดังรูปที่ 2.8



รูปที่ 2.8 โมเดลสี HSV

ที่มา http://www.medicalmac.com/img/pasted_image2.jpg

H (Hue) คือ ค่าของสีหลักหรือสีบริสุทธิ์ (Pure Color) เช่น แดง เหลือง กล่าวคือเป็นมุมตั้งแต่ 0 ถึง 360 ดังที่แสดงในตาราง

ANGLE	COLOR
0-60	RED
60-120	YELLOW
120-180	GREEN
180-240	CYAN
240-300	BLUE
300-360	MAGENTA

S (Saturation) คือ ความอิ่มตัวของสีเมื่อผสมกับแสงขาวมีค่าตั้งแต่ 0 ถึง 100 % กล่าวคือถ้า Saturation มีค่าเท่ากับ 0 จะทำให้ไม่มี Hue หมายความว่าจะเป็นสีขาวล้วนแต่ถ้ามีค่าถึง 100 % สีนั้นจะเป็นสีค่า Hue ที่ไม่มีแสงขาวผสมอยู่เลย

V (Value) คือ ความสว่างของสีที่แปรผันตาค่า Saturation มีค่าตั้งแต่ 0 ถึง 100 % เมื่อมีค่า 0 หมายถึงสีนั้นออกไปทางดำเมื่อเพิ่มค่าไปเรื่อยๆสีนั้นจะออกสว่างหรือพูดง่ายๆว่าเป็นค่าความเข้มของความสว่างนั้นเอง

2.6 โมเดลสี RGB (Red, Green และ Blue)

เป็นกระบวนการผสมสีจากแม่สี 3 ชนิด คือ สีแดง สีเขียว และสีน้ำเงิน และใช้สัดส่วนของทั้งสามสีทำให้เกิดสีต่างๆ บนจอคอมพิวเตอร์มากถึง 16.7 ล้านสีซึ่งใกล้เคียงกับสีที่ตาเรามองเห็นปกติ สีที่ได้จากการผสมสีขึ้นอยู่กับความเข้มของสี โดยถ้าสีมีความเข้มมาก เมื่อนำมาผสมกันจะทำให้เกิดเป็นสีขาว เรียกระบบสีแบบนี้ว่า Additive หรือการผสมสีแบบบวก

2.7 ทฤษฎีทาง image processing

2.7.1 Thresholds หรือตัวกำหนดความเข้มของสี ในทางการประมวลผลภาพคือ การปรับแต่งภาพให้เป็นภาพ 2 สี คือสี ขาวกับสีดำ หรือก็คือ ถ้ามีภาพอยู่ภาพหนึ่งเราจะใช้ค่า Thresholds ในการแบ่งภาพเป็น binary ซึ่งหมายถึงภาพทาง binary ที่มีเพียงค่า 2 ค่า คือ 1 และ 0

0 หมายถึง พิกเซลนั้นจะแสดงสีดำ

1 หมายถึง พิกเซลนั่นจะแสดงสีขาว

ซึ่งเมื่อเราให้ค่า Thresholds มาค่าหนึ่ง ซึ่งค่านั้นจะถูกกำหนดว่า ถ้าพิกเซลนั้นความเข้มสีเทาน้อยกว่าค่า ที่กำหนด พิกเซลนั้นจะมีค่าเป็น 0 ถ้ามากว่าค่าที่กำหนดจะมีค่าเป็น 1 รูปที่ได้จึงเป็นรูปขาว – ดำ ดังภาพตัวอย่างด่านล่าง

รูปที่ 2.9 แสดงหลังใช้ฟังก์ชั่น Threshold (ขวา) และ ก่อนใช้ ฟังก์ชั่น Threshold (ซ้าย)
ที่มา http://www.codeproject.com/Articles/31014/Image-Processing-Basics-in-C

การใช้ค่า Thresholds มีอยู่ 2 แบบ คือ Global Threshold, Local Threshold
Global Threshold คือ การใช้ค่า Thresholds ค่าเดียวทั้งภาพ

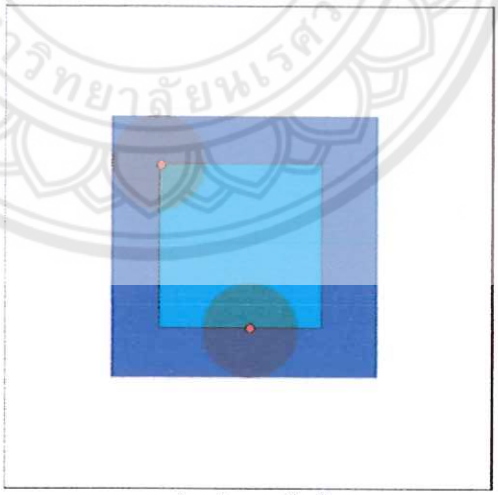
Local Threshold คือ การแบ่งภาพออกเป็นภาพย่อยๆ และกำหนดให้ภาพย่อยๆ เหล่านั้นมีค่า Threshold ของตัวเองหรือง่ายๆ ก็คือเป็น Global Threshold ของตัวเองอีกที

2.7.2 Morphological เป็นกรรมวิธีทาง image processing อย่างหนึ่ง ซึ่งประกอบด้วย

- Erosion and Dilation การกัดเซาะและการขยาย
- Opening and closing การเปิดพื้นที่ว่างภายในและการปิดพื้นที่ว่างภายใน

ในที่นี้ เราจะใช้ Erosion and Dilation การกัดเซาะและการขยาย

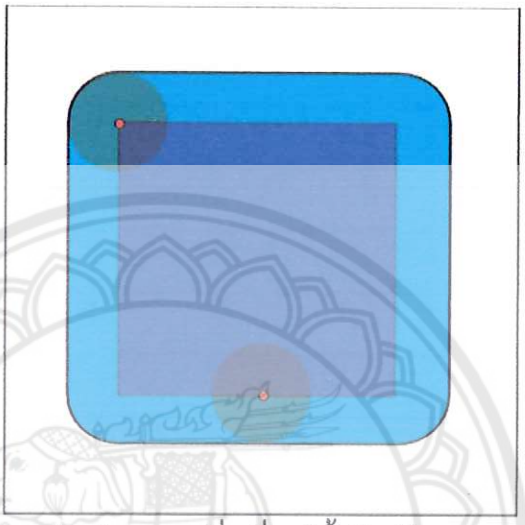
Erosion การกัดเซาะ คือการใช้ binary image มาช่วยลบส่วนเกินของรูปที่เราต้องการ จากสมการ การกัดเซาะภาพ binary A โดยองค์ประกอบโครงสร้าง B ระบุได้โดย $A \ominus B = \{z \in E | B_z \subseteq A\}$ เมื่อ B_z คือ การเลื่อนของ B ตามแนวแกน z $B_z = \{b + z | b \in B\}$, $\forall z \in E$ เมื่อองค์ประกอบโครงสร้าง B มีจุดศูนย์กลาง (เช่นเมื่อ B เป็นดิสก์หรือตาราง) และจุดศูนย์กลางนี้ตั้งอยู่บนจุดกำเนิดของ Eแล้วการกัดเซาะของ B จะสามารถเข้าใจได้เป็นสถานที่ของจุดที่เข้าถึงได้โดยทางจุดศูนย์กลางของB เมื่อ B เคลื่อนเข้ามาภายใน A เช่น การกัดเซาะของตารางด้าน 10 จุดศูนย์กลางที่จุดกำเนิด โดยแผ่นดิสก์มีรัศมี 2 และมีจุดศูนย์กลางที่จุดกำเนิดเริ่มต้นที่เป็นตารางที่ 6 ด้านมีศูนย์กลางอยู่ที่จุดเริ่มต้น ผลของการกัดเซาะของ B จะแสดงดังสมการ $A \ominus B = \bigcap_{b \in B} A_{-b}$



รูปที่ 2.10 ภาพตัวอย่างของรูปสี่เหลี่ยมสีน้ำน้ำเงินเข้ม ถูกกัดเซาะโดยวงกลม ผลที่ได้คือ รูปสี่เหลี่ยมสีฟ้า
ที่มา http://en.wikipedia.org/wiki/File:Erosion.png

Dilation การขยาย คือ การใช้ binary image มาช่วยเติมเติมส่วนที่ขาดหายของรูปที่เราต้องการจากสมการการขยายของ A โดยใช้โครงสร้าง B ระบุได้โดย $A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$

สามารถสลับที่ได้จะได้รูปสมการดังนี้ $A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$ ถ้า B มีจุดศูนย์กลางที่จุดกำเนิด แล้วขยายโดย B สามารถเข้าใจได้เป็นสถานที่ของจุดที่ครอบคลุมโดย B เมื่อจุดศูนย์กลางB เคลื่อนเข้าสู่ภายใน A การขยายสามารถเขียนได้ว่า $B^s = \{x \in E | -x \in B\}$ เมื่อ B^s สมมาตรกับ B นั่นคือ $A \oplus B = \{z \in E | (B^s)_z \cap A \neq \varnothing\}$



รูปที่ 2.11 ภาพตัวอย่างของรูปสี่เหลี่ยมสีน้ำเงินเข้ม ถูกขยายโดยวงกลม ผลที่ได้คือ รูปสี่เหลี่ยมสีฟ้าที่มีขอบมน

2.7.3 Contour คือ การหาพื้นที่ ในที่ๆ ต้องการในรูปภาพให้เด่นชัดมากขึ้น โดยกลุ่มพื้นที่นี้จะเป็นพื้นที่ในส่วนของ line contour ที่มาบรรจบกัน

2.8 การใช้งานของภาษา C++ (ภาษาโปรแกรมคอมพิวเตอร์)

C++ เป็นภาษาโปรแกรมคอมพิวเตอร์ชนิดหนึ่งที่ได้รับการพัฒนาต่อมาจากภาษา C อีกทีหนึ่ง มีการจัดชนิดข้อมูลแบบ statically และสนับสนุนการเขียนโปรแกรมที่หลากหลายเป็นวงกว้างและไม่เจาะจงกับแฟลตฟอร์มใดแฟลตฟอร์มหนึ่ง จึงเป็นภาษาทางโปรแกรมคอมพิวเตอร์ที่เหมาะสมที่จะนำไปพัฒนาต่อเป็นอย่างมาก

2.9 หลักการทำงานของ OpenCV (open source computer vision)

OpenCV เป็น Library สำหรับสำหรับสร้างฟังก์ชั่นในการโปรแกรมช่วยในการมองเห็นของคอมพิวเตอร์เพื่อใช้ในส่วนของวีดีโอและรูปภาพผ่านทางโมดูลของกล้อง ฟังก์ชั่นนี้ได้รับการพัฒนาโดยบริษัท Intel ซึ่งในที่นี้เราจะใช้ OpenCV ในการช่วยประมวลผลภาพที่เราได้รับผ่านทางกล้องเข้ามา

2.10 หลักการทำงานของซอร์ฟแวร์ CMake

CMake เป็นซอร์ฟแวร์สำหรับสร้าง build script (ไฟล์ที่นิยามวิธีการคอมไพล์และลิงค์โปรแกรมและ Library ที่เขียนด้วยภาษา C++) และสามารถใช้ร่วมกับเครื่องมือพัฒนาโปรแกรมต่างๆ ได้อาทิเช่น Microsoft Visual C++, GNU Compiler Collection, Xcode, Eclipse เป็นต้น CMake เป็นตัวแปลงการคอมไพล์ต่างๆ ที่ไม่เหมือนกันของโปรแกรมที่ใช้ภาษา C++ ของแต่ละแฟลตฟอร์ม ให้สามารถแปลงคอมไพล์ของแต่ละแฟลตฟอร์มให้เหมาะสมกับแต่ละแฟลตฟอร์มนั้น

2.11 หลักการทำงานของ WiringPi

WiringPi คือ ทางเข้า Library GPIO ที่ถูกเขียนใน C สำหรับ BCM2835 ที่ถูกใช้อยู่ใน Raspberry Pi และสามารถใช้ได้กับภาษา C และภาษา C++ และภาษาอื่นๆ WiringPi มีระบบสั่งการ GPIO (command-line utility) ที่สามารถใช้โปรแกรมหรือตั้งค่าขาต่อGPIO และเรายังสามรถเขียนหรืออ่านข้อมูลของขาต่อ หรือจะใช้ WiringPi ควบคุมขาต่อทั้งหมดผ่านทาง shell script ก็ได้

บทที่ 3

วิธีการดำเนินงาน

3.1 เลือก Hardware ที่ใช้ในการประมวลผลภาพดิจิตอล

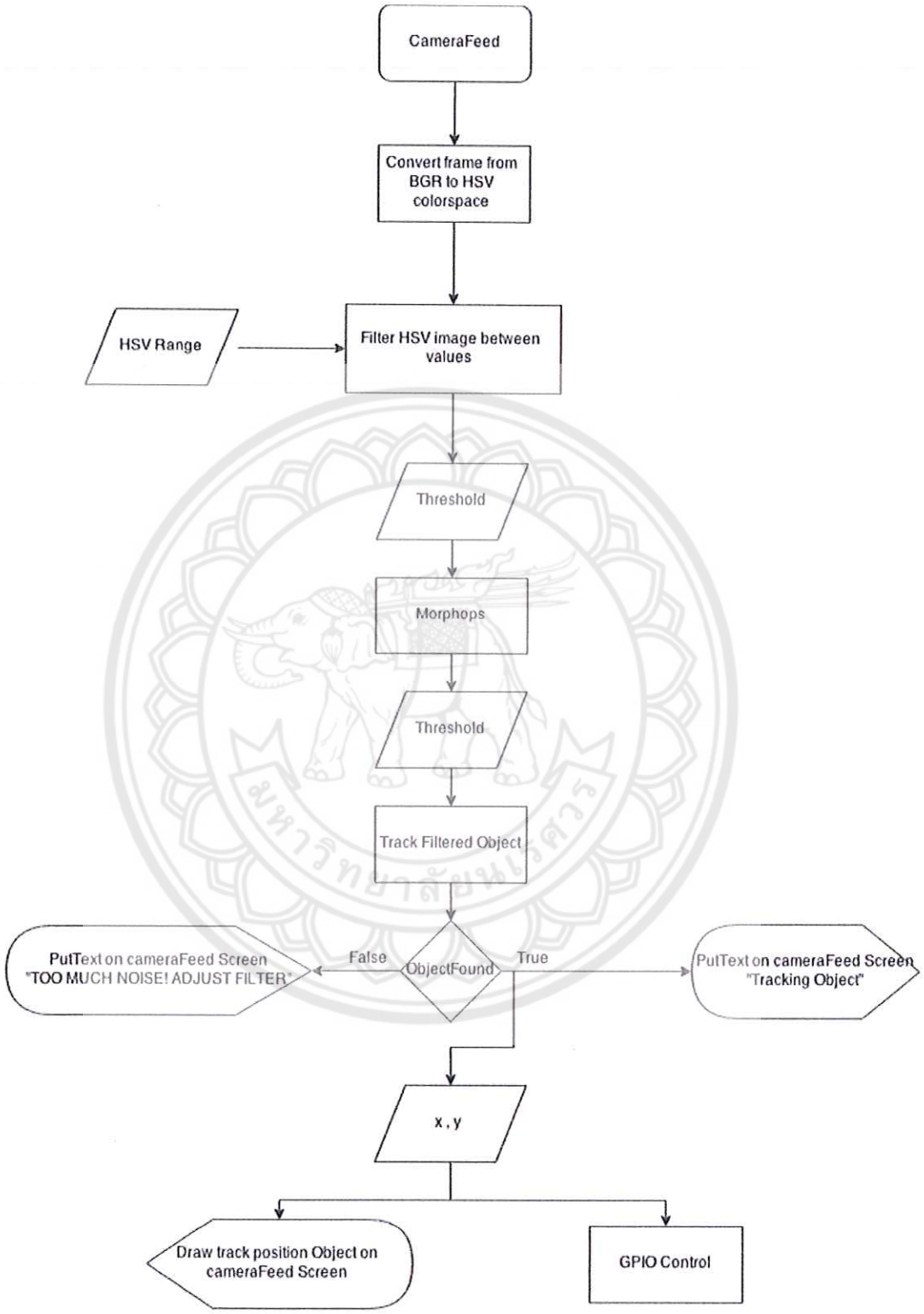
Raspberry Pi มีคุณสมบัติเป็นคอมพิวเตอร์ขนาดเล็กมีความสามารถในการประมวลผลอยู่ในระดับสูงในราคาที่เหมาะสม สร้างโปรแกรมได้ง่ายใช้คำสั่งการทำงานเช่นเดียวกับ Linux และยังสามารถนำไปใส่ในตัวหุ่นได้โดยไม่เปลืองพื้นที่ภายในของตัวหุ่นด้วย

3.2 เลือก Software ที่ใช้ในการสร้างโปรแกรมประมวลผลภาพดิจิตอล

- ใช้ภาษาในการเขียนโปรแกรมที่คุ้นเคยอย่าง C++ ที่มีพื้นฐานอยู่เล็กน้อย
- ใช้ OpenCV เป็น Function สำเร็จรูปในการประมวลผลภาพดิจิตอลสำหรับภาษา C++
- ใช้ WiringPi เป็นตัวควบคุมขา GPIO ของ Raspberry Pi เพื่อส่ง Output ออกมาเป็นสัญญาณ LED สามารถนำมาใช้กับภาษา C++ ได้

3.3 ออกแบบการทำงานในรูปแบบ Flow chart

Flow chart แสดงการทำงานกระบวนการประมวลผลภาพ 1 รูปเพื่อหา Object ภายในรูปแล้วส่งค่า y ไปสู่กระบวนการควบคุมขา GPIO

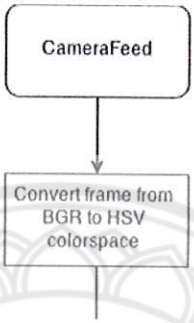


รูปที่ 3.1 แสดง Flow chart แสดงกระบวนการประมวลผลภาพของรูป 1 รูป

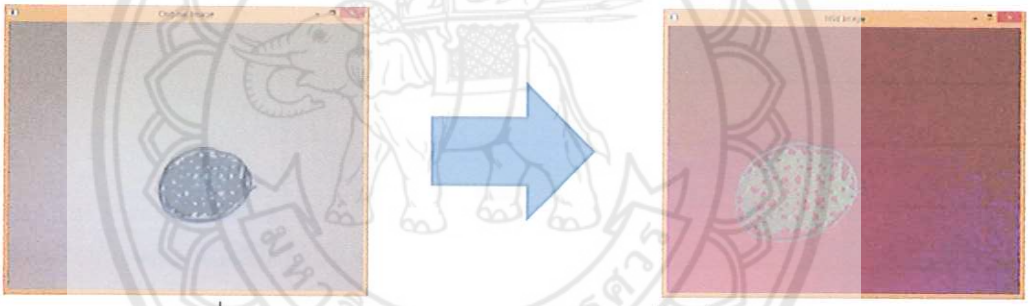
3.4 อธิบายกระบวนการย่อยของแต่ละกระบวนการใน Flow chart

3.4.1 การแปลงระบบสี RGB ไปเป็นระบบสีแบบ HSV

การแปลงสัญญาณภาพที่ได้รับเข้ามาจากจากระบบสี RGB ไปเป็นระบบสีแบบ HSV เพราะสีแบบ HSV สามารถทำการแยกสีได้ง่ายว่าแบบ RGB อย่างมาก

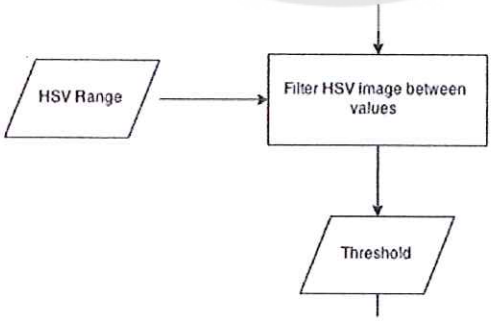


รูปที่ 3.2 Flow chart ในส่วนของการแปลงระบบสี



รูปที่ 3.3 แสดงการแปลงรูปจากระบบ RGB ไปเป็นระบบ HSV

3.4.2 เป็นการ Input ช่วงของสีที่ต้องการในระบบ HVS



รูป 3.4 แสดงส่วนการ input ค่าสีเพื่อประมวลผล

รูปที่ 3.5 แสดง range ของสีในช่วง H (Hue)

จากรูปที่ 3.5 แสดงช่วงของสีโดยเริ่มตั้งแต่ 0° – 360° การใส่ค่าจะเป็นแบบ 0-255 เช่นต้องการสีเขียว ดูจากจานสีในรูป 3.4 คือช่วงประมาณ 100° – 140° ค่าที่จะป้อนเป็น Input เป็น 71-100 โดยประมาณ

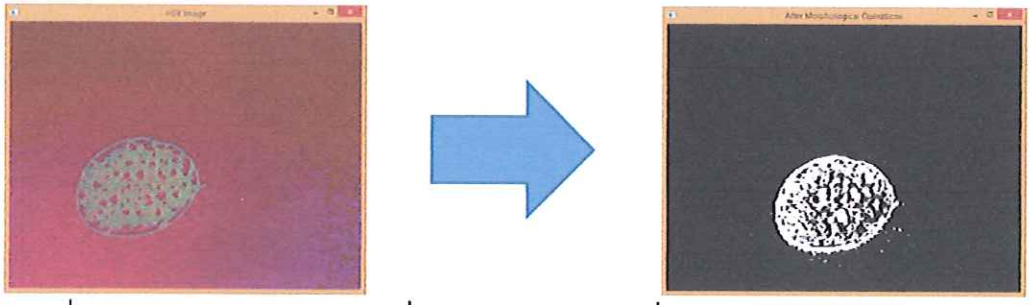
รูปที่ 3.6 ตัวอย่างแสดง range ความเข้มข้นของสีแสง S (Saturation)

จากรูป 3.6 เป็นการกำหนดความเข้มข้นของสีเป็นแบบ 0-1 การใส่ค่าจะเป็นแบบ 0-255 เช่นเดียวกัน

รูปที่ 3.7 ตัวอย่างแสดงความเข้มของแสงภายในสีแสง V (Value)

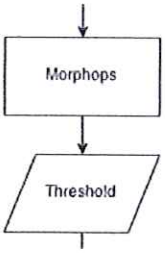
จากรูป 3.7 เป็นการกำหนดค่าความเข้มของแสงเป็นแบบ 0-255 ใส่ค่าความเข้มแสงที่ต้องการตาม 0-255 เช่นเดียวกัน

- ตัวอย่างการใส่ค่าเป็นแบบ H_MIN, H_MAX, S_MIN, S_MAX, V_MIN, V_MAX
- นำรูปภาพระบบ HSV เข้าสู่กระบวนการประมวลผลภาพเพื่อแยกสีที่ต้องการแล้วตัดสีที่ไม่ต้องการออกไป



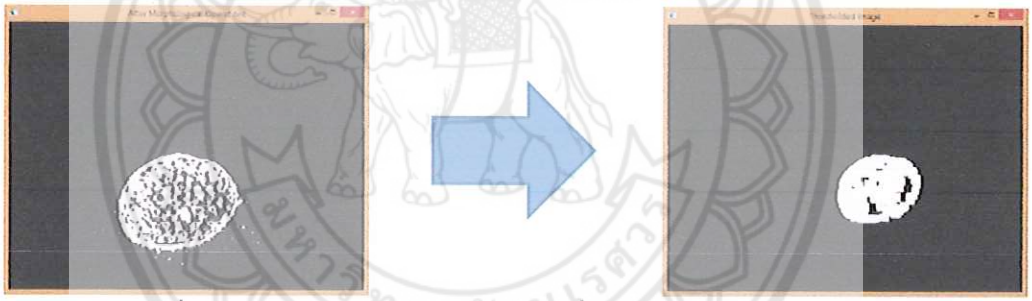
รูปที่ 3.8 แสดงกระบวนการแยกสีที่ต้องการจะได้รูปภาพที่พร้อมจะนำไประบุตำแหน่งวัตถุ

3.4.3 ทำให้วัตถุที่พบมาเพิ่มความคมชัดในการแยกแยะ



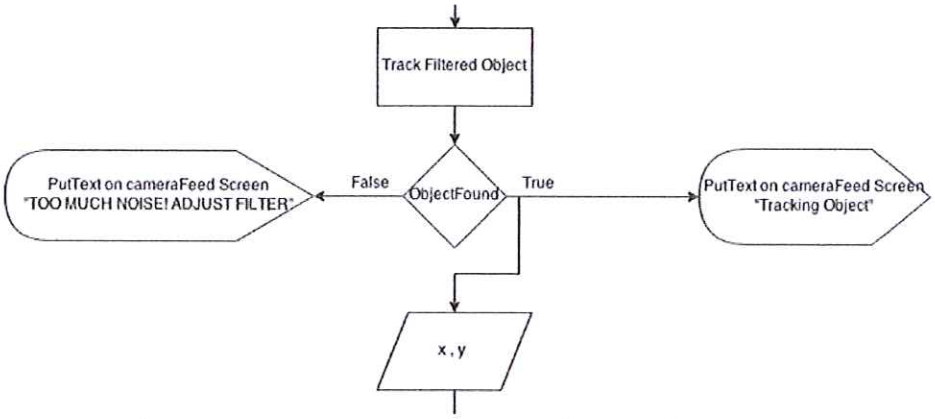
รูปที่ 3.9 แสดงส่วนการเพิ่มความคมชัดในการแยกแยะ

- สร้างองค์ประกอบโครงสร้างที่จะนำมาใช้เพื่อ "ขยาย" และ "กัดกร่อน"
- กัดกร่อนองค์ประกอบที่เลือกมีขนาดน้อยกว่า 3px * 3px
- ขยายองค์ประกอบที่มีขนาดใหญ่กว่า 3px * 3px ไปเป็นขนาด 8px * 8px เพื่อให้วัตถุมีความคมชัด



รูปที่ 3.10 แสดงรูปมาเข้ากระบวนการเพิ่มความคมชัดในการแยกแยะ

3.4.4 การประมวลผลภาพที่ปรากฏเป็นพิกัดตำแหน่ง



รูปที่ 3.11 แสดงส่วนการประมวลผลภาพที่ปรากฏเป็นพิกัดตำแหน่ง

- หารูปทรงของภาพที่ผ่านการกรองโดยใช้ฟังก์ชั่นของ OpenCV findContours ส่งออกมาเป็นพื้นที่รอบขอบวัตถุ

- ถ้าหากวัตถุมีพื้นที่น้อยกว่า 20 px * 20 px กำหนดให้สิ่งนั้นไม่ใช้วัตถุ

- ถ้าหากพื้นที่มีขนาด 3/2 ของขนาดของภาพกำหนดให้สิ่งนั้นไม่ใช้วัตถุ

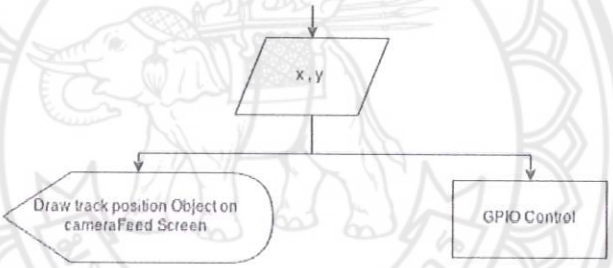
- ต้องการเพียงพื้นที่ที่มีขนาดใหญ่ที่สุดเพื่อให้สิ่งนั้นเป็นวัตถุที่ต้องการ

- นำพื้นที่ของวัตถุมาเข้าวิธีทาง Moment เพื่อที่จะหาตำแหน่งของวัตถุพิกัด x, y

- กรณีที่จะไม่พบวัตถุส่งทำการส่ง Text แสดงขึ้น "TOO MUCH NOISE! ADJUST FILTER"

- กรณีที่จะพบวัตถุส่งทำการส่ง Text แสดงขึ้น "Tracking Object" และทำการส่งค่าพิกัด x, y ไปยังกระบวนการต่อไป

3.4.5 การนำค่าพิกัด x, y เข้าสู่กระบวนการวาดเพื่อติดตามและบอกตำแหน่งวัตถุ

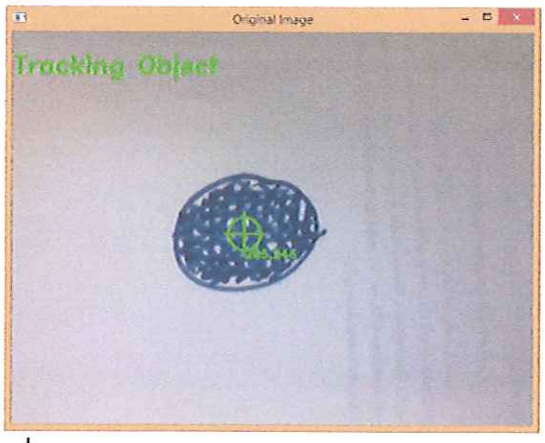


รูปที่ 3.11 แสดงส่วนการวาดการติดตามและพิกัดตำแหน่ง

- ใช้ฟังชั้นวาดของ OpenCV วาดแสดงการตรวจพอวัตถุพบรูปภาพดั้งเดิม

- นำค่า x, y ที่ได้มาใช้เป็นจุดกลางในการวาดแสดงการตรวจพอวัตถุ

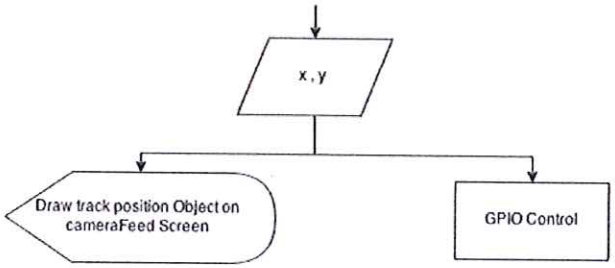
- นำค่า x, y ที่ได้มาแสดงเป็นตำแหน่งตรงที่เดียวกับวัตถุที่ตรวจพบ



รูปที่ 3.12 แสดงผลการตรวจพบวัตถุและระบุค่า x, y

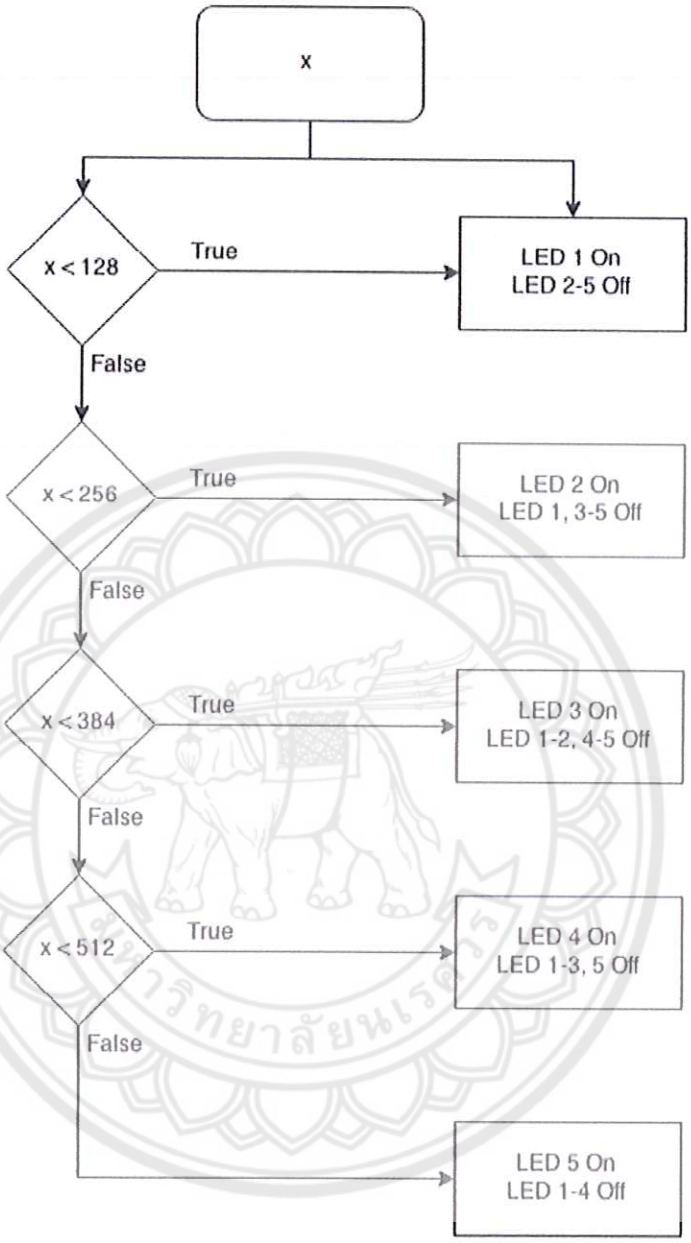
3.4.6 ความคุมการสว่างของ LED

เป็นการนำค่าพิกัด x เข้าสู่กระบวนการความคุมการสว่างของ LED ตามตำแหน่งของวัตถุ

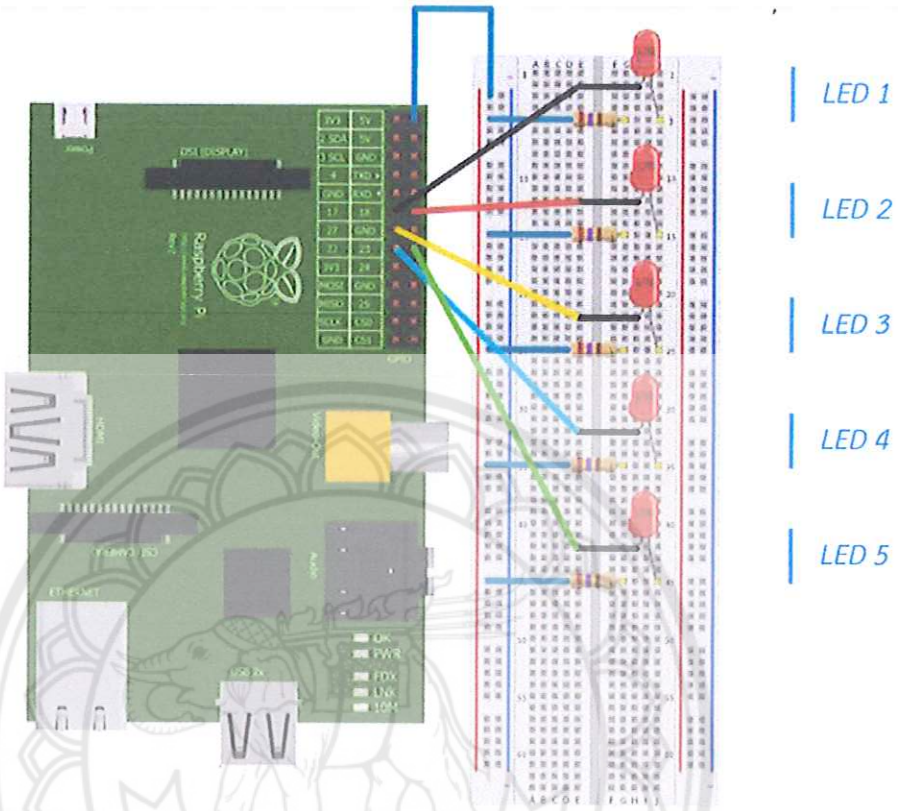


รูปที่ 3.13 แสดงส่วนการส่งค่า x ไปยังตัวควบคุม GPIO

- จากขนาดของรูปภาพที่ใช้ในการประมวลผลคือ กว้าง * ยาว = 480 * 640 px
- ทำการแบ่งความกว้างออกเป็น 5 โซน โซนละ 128 px เท่าๆกัน
- โซนซ้ายสุด 0 ถึง 128 px LED ตัวที่ 1 เปิด 2-5 ปิด
- โซนซ้าย 128 ถึง 256 px LED ตัวที่ 2 เปิด 1, 3-5 ปิด
- โซนกลาง 256 ถึง 384 px LED ตัวที่ 3 เปิด 1-2, 4-5 ปิด
- โซนขวา 348 ถึง 512 px LED ตัวที่ 4 เปิด 1-3, 5 ปิด
- โซนขวาสุด 512 ถึง 640 px LED ตัวที่ 5 เปิด 2-5 ปิด
- Flow chart แสดงการทำงานดังรูปที่ 3.14
- Raspberry Pi ที่ต่อเข้ากับวงจร LED อย่างง่ายดังรูป 3.15



รูปที่ 3.14 แสดง Flow chart ของ GPIO Controller

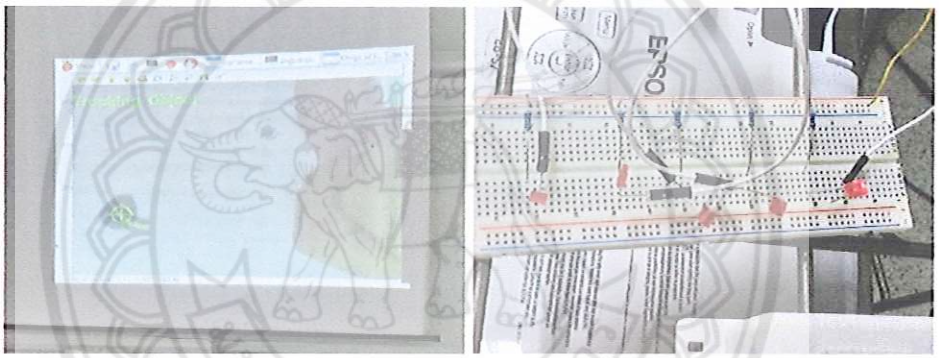


รูปที่ 3.15 แสดงการต่อ GPIO เข้ากับวงจร LED อย่างง่าย

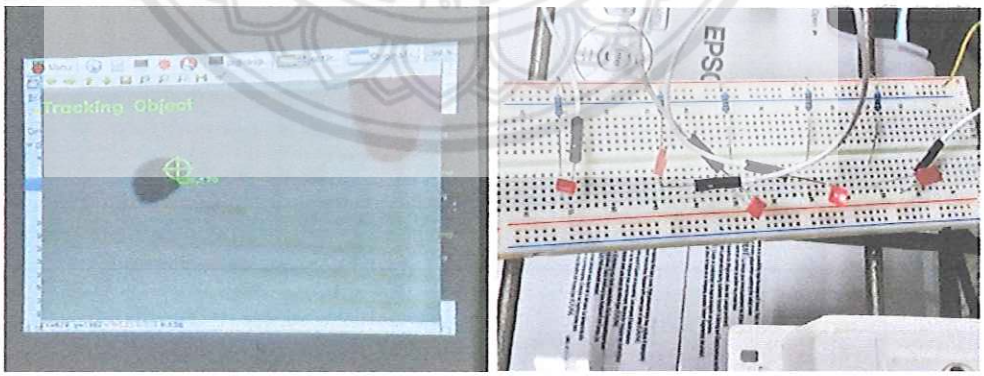
บทที่ 4

สรุปผลการทำงานของโปรแกรมที่ได้

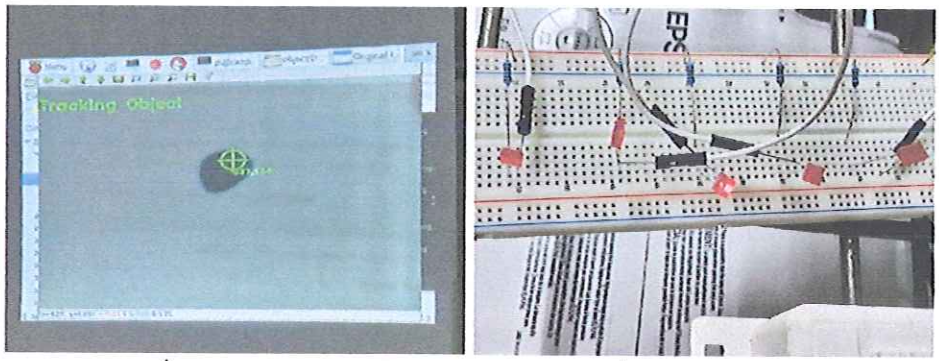
ภายในโปรแกรมนี่สามารถเลือกวัตถุที่จะทำการติดตามได้โดยทำการเลือกที่ Slides bar หรือ Fix ค่า HSV ของวัตถุเมื่อทำการเลือกได้เรียบร้อยแล้วจอแสดงผลจะปรากฏผลการติดตามตำแหน่งของวัตถุโดยทันที่แล้ว LED ทั้ง 5 จะแสดงผลตามตำแหน่งของวัตถุแบบซ้ายสุด – ซ้าย – กลาง – ขวา – ขวาสุด วีดีโอสรุปผลสามารถดูได้ที่ https://youtu.be/nXDUUxET920



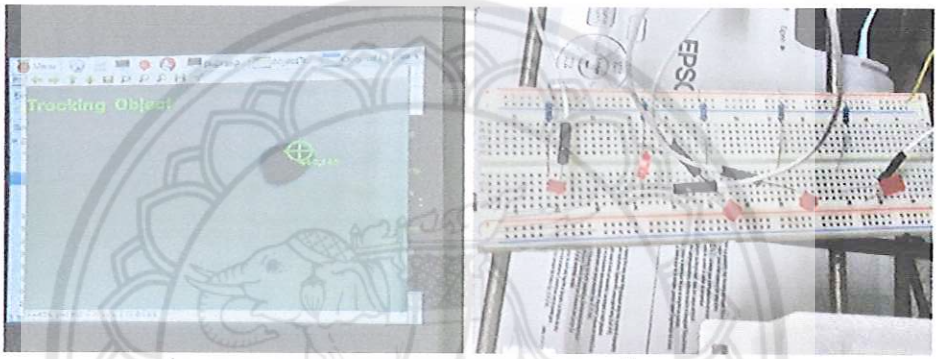
รูปที่ 4.1 แสดงตำแหน่งในรูปภาพและ LED ที่แสดงในโซนซ้ายสุด



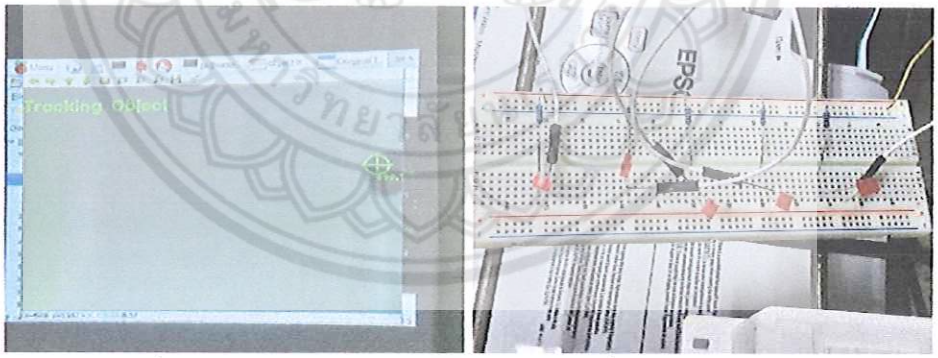
รูปที่ 4.2 แสดงตำแหน่งในรูปภาพและ LED ที่แสดงในโซนซ้าย



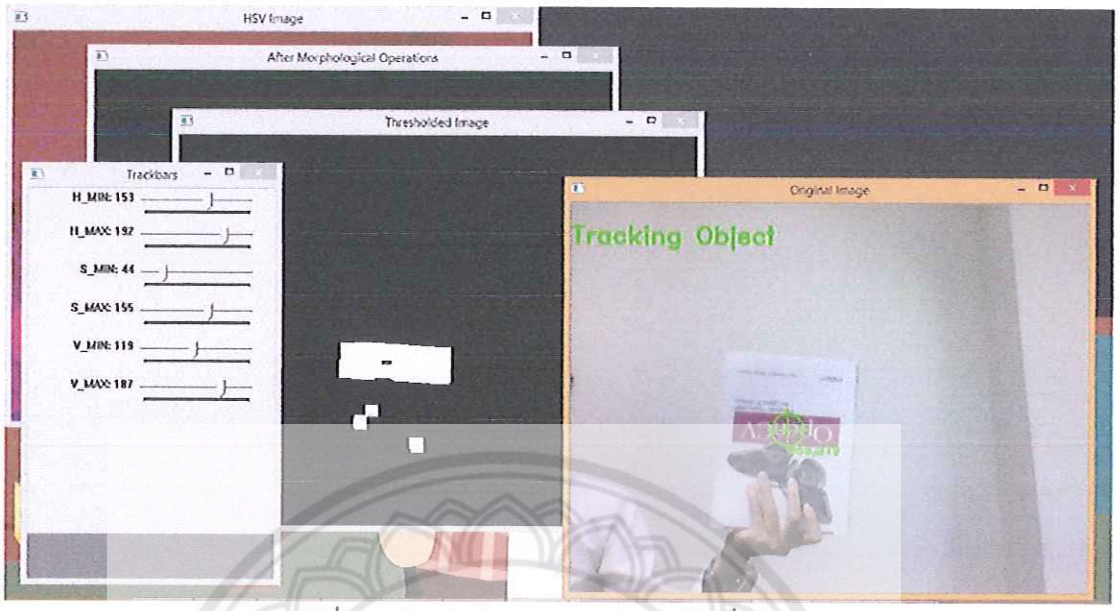
รูปที่ 4.3 แสดงตำแหน่งในรูปภาพและ LED ที่แสดงในโซนกลาง



รูปที่ 4.4 แสดงตำแหน่งในรูปภาพและ LED ที่แสดงในโซนขวา



รูปที่ 4.5 แสดงตำแหน่งในรูปภาพและ LED ที่แสดงในโซนขวาสุด



รูปที่ 4.6 GUI การทำงานของโปรแกรมที่สมบูรณ์

ภาคผนวก ก

ขั้นตอนการติดตั้ง Raspberry Pi

1. เตรียม Raspberry Pi ให้พร้อมต่อการลงระบบปฏิบัติการให้พร้อมใช้งานพัฒนาโปรแกรมและเตรียมการพัฒนาโปรแกรมบน Microsoft Windows

สายกล้องต่อไปที่ช่องที่อยู่ข้างหลังติดกับสายแลน (S5) ตามรูปที่ 1 ต่อสายสัญญาณภาพเข้ากับจอคอมพิวเตอร์ในที่นี่รูต่อสัญญาณเป็นแบบ HDMI ในส่วนจอคอมพิวเตอร์สายเป็นแบบ VGA จึงใช้ตัวแปลงสัญญาณในการต่อเข้าช่วยจำนั้นต่อ keyboard และ mouse เข้าไปที่ช่อง USB ของตัวเครื่อง

รูปแสดงการต่อกล้องและจอคอมพิวเตอร์

1.2 เตรียม SD Card ในที่ใช้ในการลงระบบปฏิบัติการ

1.2.1 ดาวโหลดไฟล์ที่ใช้ในการติดตั้ง

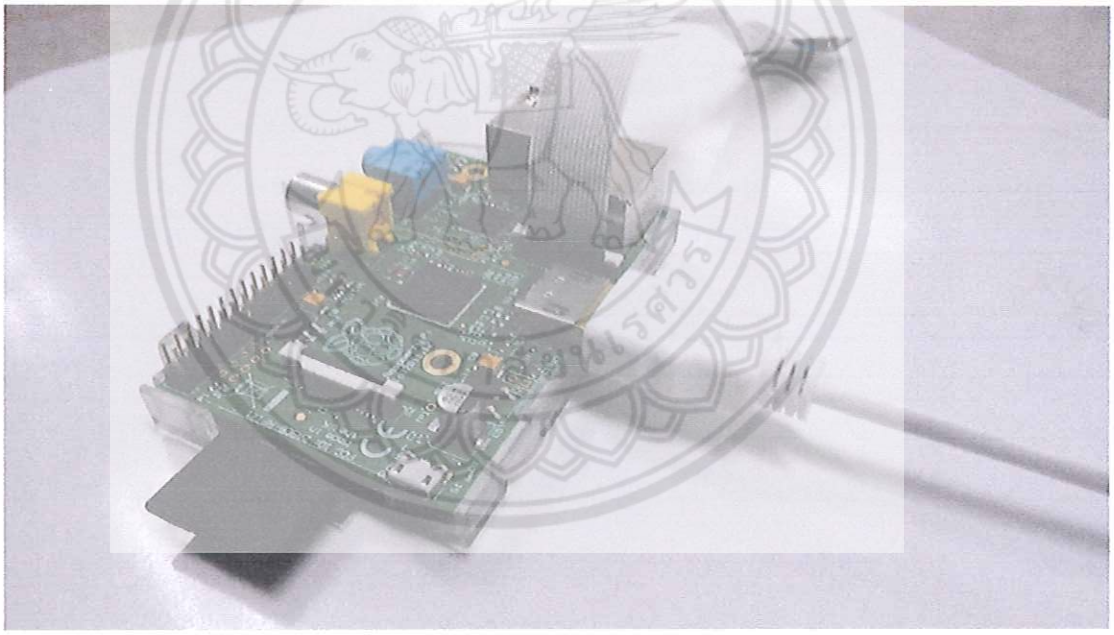
- เข้าไปที่เว็บไซต์ https://www.raspberrypi.org/downloads/
- ดาวโหลดไฟล์ที่ชื่อว่า NOOBS (Offline and network install) แบบ zip ไฟล์
- ทำการแตกไฟล์ zip ไว้ใน folder ที่ต้องการ

1.2.2 ทำการ Format SD Card

- เข้าไปที่เว็บไซต์ https://www.sdcard.org/downloads/formatter_4/
- ดาวโหลดโปรแกรม SD Formatter 4.0 สำหรับ Windows
- ทำการติดตั้งโปรแกรมนี่ลงบนคอมพิวเตอร์
- ทำการต่อ SD Card เข้าไปในคอมพิวเตอร์
- เปิดโปรแกรม SD Formatter 4.0 เลือกไดร์ฟของ SD Card แล้วทำการ Format

1.2.3 นำไฟล์ลงใน SD Card

- นำไฟล์ทั้งหมดใน folder ที่แตกแล้วของ NOOBS ไปยัง SD Card
- เมื่อนำไฟล์ทั้งเข้า SD Card เสร็จเรียบร้อยนำ SD Card นี่ไปต่อยัง Raspberry Pi



รูปแสดงการต่อ SD Card เข้ากับ Raspberry pi

1.3 การเปิดเครื่องครั้งแรกและการติดตั้งระบบปฏิบัติการ

- ทำการต่อสาย USB Power เข้ากับ Raspberry
- Raspberry pi จะทำการ boot ทันทีและจะปรากฏหน้าต่างของระบบปฏิบัติการที่สามารถติดตั้งได้ใน
- ทำการเลือก Raspbian แล้วทำการติดตั้งโดยทันที
- เมื่อทำการติดตั้งเสร็จเรียบร้อยแล้ว Raspberry Pi จำทำการโหลดหน้าต่างการตั้งค่าในครั้งแรกหลังจากติดตั้งเสร็จแล้วในหน้านี่ให้ทำการตั้งวันที่ เวลาและเปิดการใช้งาน Raspberry Pi Camera เปิดการใช้งาน SHH จากนั้นเลือกที่ Finish
- การ Login และเข้าใช้งานจะใช้ชื่อ pi และรหัส raspberry
- การใช้ GUI โดยการใช้คำสั่ง startx
- วีดีโอขั้นตอนการติดตั้งระบบปฏิบัติการสามารถดูได้ที่ https://www.raspberrypi.org/help/noobs-setup/

1.4 การติดตั้ง OpenCV ไปยัง Raspberry Pi

- ทำการต่อ internet ไปยัง Raspberry pi
- ทำการเปิด LXTerminal
- เพื่อให้แน่ใจว่า Raspberry pi อัพเดทใช้คำสั่ง "sudo apt-get update"
- ตามด้วยคำสั่ง "sudo apt-get upgrade"
- ทำการลง cmake เพื่อใช้ในการ compilationใช้คำสั่ง "sudo apt-get install build-essential cmake pkg-config"
- Java (jdk – Java Development kit) "sudo apt-get install default-jdk ant"
- Gtk development "sudo apt-get install libgtkglext1-dev"
- Bison "sudo apt-get install bison"
- v4l "sudo apt-get install v4l-utils"
- ทำการดาวโหลด OpenCV ไฟล์เราสามารถตรวจสอบ version ได้ที่ http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/ ในที่นี่เลือกใช้ version 2.4.10 โดยใช้คำสั่ง "wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.10/opencv-2.4.10.zip"
- แตก zip ไฟล์ "unzip opencv-2.4.10.zip"
- เข้าไปยัง folder ที่แตก "cd opencv-2.4.10"
- สร้าง folder build "mkdir build"
- เข้าไปยัง folder build "cd build"

- ตั้งค่าโดยใช้โปรแกรม cmake “cmake -D CMAKE_BUILD_TYPE=RELEASE -D INSTALL_C_EXAMPLES=ON –D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_OPENGL=ON -D WITH_V4L=ON –D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_TBB=ON ..”

- ขั้นตอนนี่เป็นขั้นตอนในการสร้างไฟล์ที่จะติดตั้งจะใช้เวลา 10 ถึง 12 ชั่วโมง “make”

- ถึงขั้นตอนนี้พร้อมที่ใช้ในการติดตั้งแล้ว “sudo make install”

- ออกจาก folder ปัจจุบัน “cd”

- เปิดไฟล์ opencv.conf เพื่อทำการแก้ไข “sudo nano /etc/ld.so.conf.d/opencv.conf”

- เพิ่มบรรทัดนี่เข้าไป “/usr/local/lib” ทำการ save ไฟล์

- ไปยังการตั้งค่า “sudo ldconfig”

- ทำการเปิดไฟล์ bash.bashrc “sudo nano /etc/bash.bashrc”

- ทำการเพิ่มบรรทัด “PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig” และบรรทัด “export PKG_CONFIG_PATH” แล้วทำการ save ไฟล์

- ไปที่ folder ตัวทอย่างของ C Code ที่ opencv-2.4.10/samples/c

- แล้วใส่คำสั่ง “chmod +x build_all.sh” ต่อด้วย “./build_all.sh”

- ที่นี่จะสามารถทดสอบโปรแกรมตัวอย่างจำ OpenCV ได้เช่น “./facedetect”

- วีดีโอขั้นตอนการติดตั้ง OpenCV สามารถดูได้ที่ https://youtu.be/jvFM-gIGpQQ

1.5 การติดตั้ง OpenCV ลงใน Microsoft Visual Studio

- ในที่นี่ใช้ระบบปฏิบัติการ windows 8.1 64 bit, Microsoft Visual Studio 2013, OpenCV 2.4.10

- ดาวโหลด OpenCV จาก http://sourceforge.net/projects/opencvlibrary/files/

- Save ไฟล์ลงบนไดร์ฟ C: แล้วสร้าง folder OpenCV2410

- แตกไฟล์ที่โหลดมาไปยัง C:\OpenCV2410\

- เปิด Visual Studio แล้วสร้าง new project

- เลือก Create a Visual C++ Win32 Console Application > empty project

- หลังช่อง debug ใน Win32 เลือก Configuration Manager

- ช่อง Active solution platform ให้เลือก New ช่องแรกใส่ x64 เลือก copy settings from Win32

- ใน tap Property Manager เลือก Property ของ Project ที่สร้าง

- ให้แน่ใจว่าการเปลี่ยนแปลงนี่เป็นแบบ "All Configurations" และ platform เป็น x64

- ใน C/C++ >> General ช่อง Additional Include Directories ใส่

C:\OpenCV2410\opencv\build\include

C:\OpenCV2410\opencv\build\include\opencv

C:\OpenCV2410\opencv\build\include\opencv2

- ใน Linker >> General ช่อง Additional Library Directories ใส่

C:\OpenCV2410\opencv\build\x64\vc12\lib

- ใน Linker >> Input ช่อง Additional Dependencies ใส่

opencv_calib3d2410d.lib

opencv_contrib2410d.lib

opencv_core2410d.lib

opencv_features2d2410d.lib

opencv_flann2410d.lib

opencv_gpu2410d.lib

opencv_highgui2410d.lib

opencv_imgproc2410d.lib

opencv_legacy2410d.lib

opencv_ml2410d.lib

opencv_nonfree2410d.lib

opencv_objdetect2410d.lib

opencv_ocl2410d.lib

opencv_photo2410d.lib

opencv_stitching2410d.lib

opencv_superres2410d.lib

opencv_ts2410d.lib

opencv_video2410d.lib

opencv_videostab2410d.lib

- ไปที่ช่องค้นหาของ Windows ค้นหา system environment

- กดไปที่ Environment Variables ในช่อง System variables ในช่อง path เพิ่ม

C:\OpenCV2410\opencv\build\x64\vc12\bin

- วีดีโอขั้นตอนการติดตั้ง OpenCV สำหรับ Windows 8.1 64bit สามารถดูได้ที่ https://youtu.be/e_TQ9c3n_d8

1.6 การติดตั้ง WiringPi สำหรับควบคุม GPIO PIN

- ทำการติดตั้ง Library ของ WiringPi ที่เป็นตัวควบคุมขาต่อ GPIO ของตัว Raspberry Pi ลำดับขั้นตอนการติดตั้งมีดังนี้

sudo apt-get install git-core

sudo apt-get update

sudo apt-get upgrade

git clone git://git.drogon.net/wiringPi

cd wiringPi

git pull origin

./build"

- ข้อมูลเพิ่มเติมที่ http://wiringpi.com/download-and-install/

ภาคผนวก ข

Source Code

1. นำเข้า Head file ที่จำเป็นต้องใช้งาน

```
#include <wiringPi.h>
#include <sstream>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <string>
#include <iostream>
#include <unistd.h>
#include <errno.h>
#include <cv.h>
#include "opencv2/core/core.hpp"
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"

using namespace cv;
string intToString(int number){
	std::stringstream ss;
	ss << number;
	return ss.str();
}
```

2. กำหนดค่า HSV min และ max สามารถเปลี่ยนค่าเหล่านี้ได้ที่ trackbars

```
int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
```

3. กำหนดความระเอียดของการรับภาพ

```
const int FRAME_WIDTH = 640;
const int FRAME_HEIGHT = 480;
```

4. กำหนดจำนวนสูงสุดของวัตถุที่เจอ

```
const int MAX_NUM_OBJECTS=50;
```

5. กำหนดค่าสูงสุดและต่ำสุดของพื้นที่วัตถุที่เจอ

```
const int MIN_OBJECT_AREA = 20*20;
const int MAX_OBJECT_AREA = FRAME_HEIGHT*FRAME_WIDTH/1.5;
```

6. กำหนดชื่อของหน้าต่างที่จำเป็น

```
const string windowName = "Original Image";
const string windowName1 = "HSV Image";
const string windowName2 = "Thresholded Image";
const string windowName3 = "After Morphological Operations";
const string trackbarWindowName = "Trackbars";
```

7. Function นี่สามารถทราบตำแหน่งของ HSV min max เพื่อเปลี่ยนค่าได้

```
void on_trackbar( int, void* )
{//This function gets called whenever a
	// trackbar position is changed

}
```

8. Function สร้าง Trackbars

```
void createTrackbars(){
	//create window for trackbars
	namedWindow(trackbarWindowName,0);
	//create memory to store trackbar name on window
	char TrackbarName[50];
	sprintf( TrackbarName, "H_MIN", H_MIN);
	sprintf( TrackbarName, "H_MAX", H_MAX);
	sprintf( TrackbarName, "S_MIN", S_MIN);
	sprintf( TrackbarName, "S_MAX", S_MAX);
	sprintf( TrackbarName, "V_MIN", V_MIN);
	sprintf( TrackbarName, "V_MAX", V_MAX);
	//create trackbars and insert them into window
	//3 parameters are: the address of the variable that is changing when
the trackbar is moved(eg.H_LOW),
	//the max value the trackbar can move (eg. H_HIGH),
	//and the function that is called whenever the trackbar is moved(eg.
on_trackbar)
	createTrackbar( "H_MIN", trackbarWindowName, &H_MIN, H_MAX, on_trackbar
);
    createTrackbar( "H_MAX", trackbarWindowName, &H_MAX, H_MAX, on_trackbar );
    createTrackbar( "S_MIN", trackbarWindowName, &S_MIN, S_MAX, on_trackbar );
    createTrackbar( "S_MAX", trackbarWindowName, &S_MAX, S_MAX, on_trackbar );
    createTrackbar( "V_MIN", trackbarWindowName, &V_MIN, V_MAX, on_trackbar );
    createTrackbar( "V_MAX", trackbarWindowName, &V_MAX, V_MAX, on_trackbar );
}
```

9. Function การวาดตำแหน่งลงบนวัตถุที่เจอและควบคุม LED

```
void drawObject(int x, int y,Mat &frame){
	//use some of the openCV drawing functions to draw crosshairs
	//on your tracked image!
	circle(frame,Point(x,y),20,Scalar(0,255,0),2);
    if(y-25>0)
    line(frame,Point(x,y),Point(x,y-25),Scalar(0,255,0),2);
    else line(frame,Point(x,y),Point(x,0),Scalar(0,255,0),2);
    if(y+25<FRAME_HEIGHT)
    line(frame,Point(x,y),Point(x,y+25),Scalar(0,255,0),2);
    else line(frame,Point(x,y),Point(x,FRAME_HEIGHT),Scalar(0,255,0),2);
    if(x-25>0)
    line(frame,Point(x,y),Point(x-25,y),Scalar(0,255,0),2);
```

```
      else line(frame,Point(x,y),Point(0,y),Scalar(0,255,0),2);
      if(x+25<FRAME_WIDTH)
      line(frame,Point(x,y),Point(x+25,y),Scalar(0,255,0),2);
      else line(frame,Point(x,y),Point(FRAME_WIDTH,y),Scalar(0,255,0),2);
         //GPIO LED Controllor
         //0 = LED ON
         //1 = LED OFF
      if (x < 128) {
          digitalWrite (0, 0);
          digitalWrite (1, 1);
          digitalWrite (2, 1);
          digitalWrite (3, 1);
          digitalWrite (4, 1);
      }
      else if (x < 128*2 ) {
          digitalWrite (0, 1);
          digitalWrite (1, 0);
          digitalWrite (2, 1);
          digitalWrite (3, 1);
          digitalWrite (4, 1);
      }
      else if (x < 128*3 ){
          digitalWrite (0, 1);
          digitalWrite (1, 1);
          digitalWrite (2, 0);
          digitalWrite (3, 1);
          digitalWrite (4, 1);
      }
      else if (x < 128*4 ){
          digitalWrite (0, 1);
          digitalWrite (1, 1);
          digitalWrite (2, 1);
          digitalWrite (3, 0);
          digitalWrite (4, 1);
      }
      else if (x < 128*5 ){
          digitalWrite (0, 1);
          digitalWrite (1, 1);
          digitalWrite (2, 1);
          digitalWrite (3, 1);
          digitalWrite (4, 0);
      }
         //Text Position x, y
         putText(frame,intToString(x)+","+intToString(y),Point(x,y+30),1,1,Scalar
(0,255,0),2);

}
```

10. Function Morphology Transformations

```
void morphOps(Mat &thresh){

       //create structuring element that will be used to "dilate" and "erode"
image.
       //the element chosen here is a 3px by 3px rectangle

       Mat erodeElement = getStructuringElement( MORPH_RECT,Size(3,3));
    //dilate with larger element so make sure object is nicely visible
       Mat dilateElement = getStructuringElement( MORPH_RECT,Size(8,8));
```

```
        erode(thresh,thresh,erodeElement);
        erode(thresh,thresh,erodeElement);


        dilate(thresh,thresh,dilateElement);
        dilate(thresh,thresh,dilateElement);

}
```

11. Function การกรองวัตถุที่จะ Track

```
void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed){

        Mat temp;
        threshold.copyTo(temp);
        //these two vectors needed for output of findContours
        vector< vector<Point> > contours;
        vector<Vec4i> hierarchy;
        //find contours of filtered image using openCV findContours function
        findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPL
E );
        //use moments method to find our filtered object
        double refArea = 0;
        bool objectFound = false;
        if (hierarchy.size() > 0) {
                int numObjects = hierarchy.size();
         //if number of objects greater than MAX_NUM_OBJECTS we have a noisy
filter
         if(numObjects<MAX_NUM_OBJECTS){
                        for (int index = 0; index >= 0; index =
hierarchy[index][0]) {

                                Moments moment = moments((cv::Mat)contours[index]);
                                double area = moment.m00;

                                //if the area is less than 20 px by 20px then it is
probably just noise
                                //if the area is the same as the 3/2 of the image
size, probably just a bad filter
                                //we only want the object with the largest area so
we safe a reference area each
                                //iteration and compare it to the area in the next
iteration.
                        if(area>MIN_OBJECT_AREA && area<MAX_OBJECT_AREA &&
area>refArea){
                                        x = moment.m10/area;
                                        y = moment.m01/area;
                                        objectFound = true;
                                        refArea = area;
                                }else objectFound = false;


                        }
                        //let user know you found an object
                        if(objectFound ==true){
                                putText(cameraFeed,"Tracking
Object",Point(0,50),2,1,Scalar(0,255,0),2);
```

```
                        //draw object location on screen
                        drawObject(x,y,cameraFeed);}

        }else {putText(cameraFeed,"TOO MUCH NOISE! ADJUST
FILTER",Point(0,50),1,2,Scalar(0,0,255),2);

        }

       }
}
```

12. เขียน Function หลักในการทำงานและเรียกใช้ Function ย่อยที่สร้าง

```
int main(int argc, char* argv[])
{
       //some boolean variables for different functionality within this
       //program
       //Import GPIO PIN
    if (wiringPiSetup () == -1)
      exit (1);
    pinMode(0, OUTPUT);
    pinMode(1, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
       //set default all LED OFF
    digitalWrite (0, 1);
    digitalWrite (1, 1);
    digitalWrite (2, 1);
    digitalWrite (3, 1);
    digitalWrite (4, 1);
       //set use function if you need
       bool trackObjects = true;
       bool useMorphOps = true;
       //Matrix to store each frame of the webcam feed
       Mat cameraFeed;
       //matrix storage for HSV image
       Mat HSV;
       //matrix storage for binary threshold image
       Mat threshold;
       Mat threshold2;
       //x and y values for the location of the object
       int x=0, y=0;
       //create slider bars for HSV filtering
       createTrackbars();
       //video capture object to acquire webcam feed
       VideoCapture capture;
       //open capture object at location zero (default location for webcam)
       capture.open(0);
       //set height and width of capture frame
       capture.set(CV_CAP_PROP_FRAME_WIDTH,FRAME_WIDTH);
       capture.set(CV_CAP_PROP_FRAME_HEIGHT,FRAME_HEIGHT);
       //start an infinite loop where webcam feed is copied to cameraFeed
matrix
       //all of our operations will be performed within this loop
       while(1){
              //store image to matrix
              capture.read(cameraFeed);
              //convert frame from BGR to HSV colorspace
```

```
                cvtColor( cameraFeed, HSV, COLOR_BGR2HSV);
                //filter HSV image between values and store filtered image to
                //threshold matrix
                //in Scalar(H_MIN,S_MIN,V_MIN) you can set fix HSV min range here
                //in Scalar(H_MAX,S_MAX,V_MAX) you can set fix HSV max range here

        inRange(HSV,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX),threshol
d);
                //stock threshold image After Morphological Operations
                threshold.copyTo(Mat threshold2);
                //perform morphological operations on thresholded image to
eliminate noise
                //and emphasize the filtered object(s)
                if(useMorphOps)
                morphOps(threshold);
                //pass in thresholded frame to our object tracking function
                //this function will return the x and y coordinates of the
                //filtered object
         if(trackObjects){
                        trackFilteredObject(x,y,threshold,cameraFeed);
             }

                //show frames
                imshow(windowName2,threshold);
                imshow(windowName,cameraFeed);
         imshow(windowName1,HSV);
                //Delete // from "//imshow(windowName3,threshold2);" for see
threshold image after morphological operations
                //imshow(windowName3,threshold2);


                //delay 30ms so that screen can refresh.
                //image will not appear without this waitKey() command
                waitKey(30);
        }


        return 0;
}
```

ภาคผนวก ค

Compiler File

การ Compiler Code บน Raspberry Pi โดยใช้ cmake

- เพื่อให้ง่ายต่อการ compiler จึงต้องทำการสร้าง folder ไว้บน Desktop ของ Raspbian ในที่นี่เราใช้ชื่อ "ObjectTracking"

- นำไฟล์ของ C++ ที่เขียนแล้วมาไว้ใน folder ที่สร้างขึ้นมาไฟล์ C++ ในที่นี่มีชื่อว่า "ObjectTracking.cpp"

- สร้างอีกหนึ่งไฟล์ซึ่งไฟล์นี่จะเป็นตัวกำหนดการ compiler code สามารถสร้างโดย Notepad ของ Windows เองให้ชื่อว่า "CmakeLists.txt" เท่านั้นภายในไฟล์นี่ให้เขียน

"CMAKE_MINIMUM_REQUIRED(VERSION 2.6)
project(objectTracking)
SET(CMAKE_CXX_FLAGS "-pthread -I/usr/local/include -L/usr/local/lib -lwiringPi")
find_package(OpenCV REQUIRED)
add_executable(objectTracking objectTracking.cpp)
target_link_libraries(objectTracking ${OpenCV_LIBS})"

- นำไฟล์ "CmakeLists.txt" ไปไว้ในที่เดียวกันกับไฟล์ "ObjectTracking.cpp"

- ทำการเปิด Terminal แล้วใช้คำสั่ง "cd Desktop/ObjectTracking" เพื่อที่จะกระทำการภายใน folder "ObjectTracking"

- ใช้คำสั่ง "cmake ." เพื่อบอกให้ CMake สร้าง build script โดยอ่าน CMakeLists.txt จากไดเรคทอรีปัจจุบัน

- cmake จะทำการสร้างไฟล์ที่สำคัญเพื่อใช้ในคำสั่งต่อไป

- ใช้คำสั่ง "make" จะได้ไฟล์ executable ชื่อ ObjectTracking เพื่อให้ได้โปรแกรมที่พร้อมใช้งานบน Raspbian

- การสั่งเปิดโปรแกรมนั้นจำเป็นจะต้องอยู่ในไดเรคทอรีที่โปรแกรมนั้นอยู่แล้วใช้คำสั่ง "sudo ./ObjectTracking" โปรแกรมจะถูกเปิดขึ้นมา

***หมายเหตุ ในการเปิด Raspberry Pi ครั้งแรกนั้นต้องทำการเรียกคำสั่ง "sudo modprobe bcm2835-v4l2" ก่อนที่จะเปิด "./ObjectTracking" แค่ครั้งเดียวก็เพียงพอถ้าหายปิดเครื่องหรือรีบูตจะต้องเรียกคำสั่ง "sudo modprobe bcm2835-v4l2" อีก

ภาคผนวก ง

แสดง soured code ของโปรแกรมฉบับเต็ม

```
#include <wiringPi.h>
#include <sstream>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <string>
#include <iostream>
#include <unistd.h>
#include <errno.h>
#include <cv.h>
#include "opencv2/core/core.hpp"
#include "opencv2/video/tracking.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"

using namespace cv;
//initial min and max HSV filter values.
//these will be changed using trackbars
//system(sudo modprobe bcm2835-v4l2);
int H_MIN = 0;
int H_MAX = 256;
int S_MIN = 0;
int S_MAX = 256;
int V_MIN = 0;
int V_MAX = 256;
//default capture width and height
const int FRAME_WIDTH = 640;
const int FRAME_HEIGHT = 480;
//max number of objects to be detected in frame
const int MAX_NUM_OBJECTS=50;
//minimum and maximum object area
const int MIN_OBJECT_AREA = 20*20;
const int MAX_OBJECT_AREA = FRAME_HEIGHT*FRAME_WIDTH/1.5;
//names that will appear at the top of each window
const string windowName = "Original Image";
const string windowName1 = "HSV Image";
const string windowName2 = "Thresholded Image";
const string windowName3 = "After Morphological Operations";
const string trackbarWindowName = "Trackbars";

void on_trackbar( int, void* )
{//This function gets called whenever a
	// trackbar position is changed

}
string intToString(int number){
	std::stringstream ss;
	ss << number;
	return ss.str();
}
void createTrackbars(){
	//create window for trackbars
	namedWindow(trackbarWindowName,0);
	//create memory to store trackbar name on window
	char TrackbarName[50];
```

```
        sprintf( TrackbarName, "H_MIN", H_MIN);
        sprintf( TrackbarName, "H_MAX", H_MAX);
        sprintf( TrackbarName, "S_MIN", S_MIN);
        sprintf( TrackbarName, "S_MAX", S_MAX);
        sprintf( TrackbarName, "V_MIN", V_MIN);
        sprintf( TrackbarName, "V_MAX", V_MAX);
        //create trackbars and insert them into window
        //3 parameters are: the address of the variable that is changing when
the trackbar is moved(eg.H_LOW),
        //the max value the trackbar can move (eg. H_HIGH),
        //and the function that is called whenever the trackbar is moved(eg.
on_trackbar)
        createTrackbar( "H_MIN", trackbarWindowName, &H_MIN, H_MAX, on_trackbar
);
    createTrackbar( "H_MAX", trackbarWindowName, &H_MAX, H_MAX, on_trackbar );
    createTrackbar( "S_MIN", trackbarWindowName, &S_MIN, S_MAX, on_trackbar );
    createTrackbar( "S_MAX", trackbarWindowName, &S_MAX, S_MAX, on_trackbar );
    createTrackbar( "V_MIN", trackbarWindowName, &V_MIN, V_MAX, on_trackbar );
    createTrackbar( "V_MAX", trackbarWindowName, &V_MAX, V_MAX, on_trackbar );
}
void drawObject(int x, int y,Mat &frame){
        //use some of the openCV drawing functions to draw crosshairs
        //on your tracked image!
        circle(frame,Point(x,y),20,Scalar(0,255,0),2);
    if(y-25>0)
    line(frame,Point(x,y),Point(x,y-25),Scalar(0,255,0),2);
    else line(frame,Point(x,y),Point(x,0),Scalar(0,255,0),2);
    if(y+25<FRAME_HEIGHT)
    line(frame,Point(x,y),Point(x,y+25),Scalar(0,255,0),2);
    else line(frame,Point(x,y),Point(x,FRAME_HEIGHT),Scalar(0,255,0),2);
    if(x-25>0)
    line(frame,Point(x,y),Point(x-25,y),Scalar(0,255,0),2);
    else line(frame,Point(x,y),Point(0,y),Scalar(0,255,0),2);
    if(x+25<FRAME_WIDTH)
    line(frame,Point(x,y),Point(x+25,y),Scalar(0,255,0),2);
    else line(frame,Point(x,y),Point(FRAME_WIDTH,y),Scalar(0,255,0),2);
        //GPIO LED Controllor
        //0 = LED ON
        //1 = LED OFF
    if (x < 128) {
        digitalWrite (0, 0);
        digitalWrite (1, 1);
        digitalWrite (2, 1);
        digitalWrite (3, 1);
        digitalWrite (4, 1);
    }
    else if (x < 128*2 ) {
        digitalWrite (0, 1);
        digitalWrite (1, 0);
        digitalWrite (2, 1);
        digitalWrite (3, 1);
        digitalWrite (4, 1);
    }
    else if (x < 128*3 ){
        digitalWrite (0, 1);
        digitalWrite (1, 1);
        digitalWrite (2, 0);
        digitalWrite (3, 1);
        digitalWrite (4, 1);
```

```
    }
    else if (x < 128*4 ){
        digitalWrite (0, 1);
        digitalWrite (1, 1);
        digitalWrite (2, 1);
        digitalWrite (3, 0);
        digitalWrite (4, 1);
    }
    else if (x < 128*5 ){
        digitalWrite (0, 1);
        digitalWrite (1, 1);
        digitalWrite (2, 1);
        digitalWrite (3, 1);
        digitalWrite (4, 0);
    }
       //Text Position x, y
       putText(frame,intToString(x)+","+intToString(y),Point(x,y+30),1,1,Scalar
(0,255,0),2);

}

void morphOps(Mat &thresh){

       //create structuring element that will be used to "dilate" and "erode"
image.
       //the element chosen here is a 3px by 3px rectangle

       Mat erodeElement = getStructuringElement( MORPH_RECT,Size(3,3));
    //dilate with larger element so make sure object is nicely visible
       Mat dilateElement = getStructuringElement( MORPH_RECT,Size(8,8));

       erode(thresh,thresh,erodeElement);
       erode(thresh,thresh,erodeElement);


       dilate(thresh,thresh,dilateElement);
       dilate(thresh,thresh,dilateElement);

}
void trackFilteredObject(int &x, int &y, Mat threshold, Mat &cameraFeed){

       Mat temp;
       threshold.copyTo(temp);
       //these two vectors needed for output of findContours
       vector< vector<Point> > contours;
       vector<Vec4i> hierarchy;
       //find contours of filtered image using openCV findContours function
       findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_APPROX_SIMPL
E );
       //use moments method to find our filtered object
       double refArea = 0;
       bool objectFound = false;
       if (hierarchy.size() > 0) {
              int numObjects = hierarchy.size();
        //if number of objects greater than MAX_NUM_OBJECTS we have a noisy
filter
        if(numObjects<MAX_NUM_OBJECTS){
                    for (int index = 0; index >= 0; index =
hierarchy[index][0]) {
```

```
                        Moments moment = moments((cv::Mat)contours[index]);
                        double area = moment.m00;

                        //if the area is less than 20 px by 20px then it is
probably just noise
                        //if the area is the same as the 3/2 of the image
size, probably just a bad filter
                        //we only want the object with the largest area so
we safe a reference area each
                        //iteration and compare it to the area in the next
iteration.
                if(area>MIN_OBJECT_AREA && area<MAX_OBJECT_AREA &&
area>refArea){
                                x = moment.m10/area;
                                y = moment.m01/area;
                                objectFound = true;
                                refArea = area;
                        }else objectFound = false;
                }
                //let user know you found an object
                if(objectFound ==true){
                        putText(cameraFeed,"Tracking
Object",Point(0,50),2,1,Scalar(0,255,0),2);
                        //draw object location on screen
                        drawObject(x,y,cameraFeed);}

        }else {putText(cameraFeed,"TOO MUCH NOISE! ADJUST
FILTER",Point(0,50),1,2,Scalar(0,0,255),2);

        }

       }
}

int main(int argc, char* argv[])
{
       //some boolean variables for different functionality within this
       //program
       //Import GPIO PIN
    if (wiringPiSetup () == -1)
      exit (1);
    pinMode(0, OUTPUT);
    pinMode(1, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
       //set default all LED OFF
    digitalWrite (0, 1);
    digitalWrite (1, 1);
    digitalWrite (2, 1);
    digitalWrite (3, 1);
    digitalWrite (4, 1);
       //set use function if you need
       bool trackObjects = true;
       bool useMorphOps = true;
       //Matrix to store each frame of the webcam feed
       Mat cameraFeed;
       //matrix storage for HSV image
       Mat HSV;
```

```
	//matrix storage for binary threshold image
	Mat threshold;
	Mat threshold2;
	//x and y values for the location of the object
	int x=0, y=0;
	//create slider bars for HSV filtering
	createTrackbars();
	//video capture object to acquire webcam feed
	VideoCapture capture;
	//open capture object at location zero (default location for webcam)
	capture.open(0);
	//set height and width of capture frame
	capture.set(CV_CAP_PROP_FRAME_WIDTH,FRAME_WIDTH);
	capture.set(CV_CAP_PROP_FRAME_HEIGHT,FRAME_HEIGHT);
	//start an infinite loop where webcam feed is copied to cameraFeed
matrix
	//all of our operations will be performed within this loop
	while(1){
		//store image to matrix
		capture.read(cameraFeed);
		//convert frame from BGR to HSV colorspace
		cvtColor( cameraFeed, HSV, COLOR_BGR2HSV);
		//filter HSV image between values and store filtered image to
		//threshold matrix
		//in Scalar(H_MIN,S_MIN,V_MIN) you can set fix HSV min range here
		//in Scalar(H_MAX,S_MAX,V_MAX) you can set fix HSV max range here

	inRange(HSV,Scalar(H_MIN,S_MIN,V_MIN),Scalar(H_MAX,S_MAX,V_MAX),threshol
d);
		//stock threshold image After Morphological Operations
		threshold.copyTo(Mat threshold2);
		//perform morphological operations on thresholded image to
eliminate noise
		//and emphasize the filtered object(s)
		if(useMorphOps)
		morphOps(threshold);
		//pass in thresholded frame to our object tracking function
		//this function will return the x and y coordinates of the
		//filtered object
	 if(trackObjects){
			trackFilteredObject(x,y,threshold,cameraFeed);
		}

		//show frames
		imshow(windowName2,threshold);
		imshow(windowName,cameraFeed);
	 imshow(windowName1,HSV);
		//Delete // from "//imshow(windowName3,threshold2);" for see
threshold image after morphological operations
		//imshow(windowName3,threshold2);

		//delay 30ms so that screen can refresh.
		//image will not appear without this waitKey() command
		waitKey(30);
	}

	return 0;
}
```

ประวัติผู้จัดทำโครงงาน

ชื่อ-นามสกุล นายเจตน์จักร เนตรสุวรรณ
วันเดือนปีเกิด วันอังคาร ที่ 6เมษายน 2536
ที่อยู่ บ้านเลขที่ 28/20 ถนนวังคาง ตำบลในเมืองอำเภอเมือง จังหวัดกำแพงเพชร 62000
ประวัติการศึกษา
-พ.ศ.2553 จบชั้นมัธยมศึกษาตอนปลายที่โรงเรียนกำแพงเพชรพิทยาคม
-ปัจจุบันกำลังศึกษาระดับปริญญาตรี ชั้นปีที่ 4 สาขาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ชื่อ-นามสกุล นายสุทธิรงค์ อยู่เอม
วันเดือนปีเกิด วันอาทิตย์ ที่ 16 พฤษภาคม 2536
ที่อยู่ บ้านเลขที่ 14/14 หมู่ที่ 6 ต.สระแก้ว อ.เมือง จ.กำแพงเพชร 62000
ประวัติการศึกษา
-พ.ศ.2553 จบชั้นมัธยมศึกษาตอนปลายที่โรงเรียนกำแพงเพชรพิทยาคม
-ปัจจุบันกำลังศึกษาระดับปริญญาตรี ชั้นปีที่ 4 สาขาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ชื่อ-นามสกุล นายจิณณวัตร เรืองนิ่ม
วันเดือนปีเกิด วันอาทิตย์ ที่ 29 พฤศจิกายน 2535
ที่อยู่ บ้านเลขที่ 2/160 ถ.สุดบรรทัด ต.แก่งคอย อ.แก่งคอย จ. สระบุรี 18110
ประวัติการศึกษา
-พ.ศ.2553 จบชั้นมัธยมศึกษาตอนปลายที่โรงเรียนพิบูลวิทยาลัยจังหวัดลพบุรี
-ปัจจุบันกำลังศึกษาระดับปริญญาตรี ชั้นปีที่ 4 สาขาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร