AMAS 2016

FRAB#1 TEAM

FIBO KMUTT

Members:

1. Jettanan Homchanthanakul
2. Nutthaya Hankla
3. Sirawat Soksawatmakin

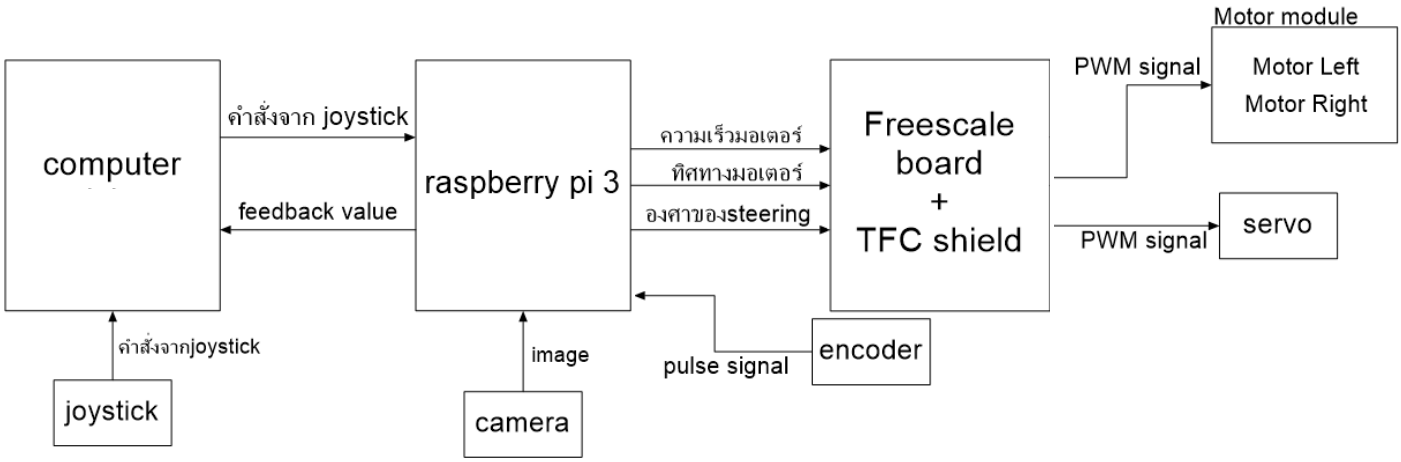
Advisor :

Dr. Thavida Maneewarn

แผนและตารางการดำเนินงาน

ขั้นตอนการดำเนินงาน	การดำเนินงาน						
	2559					2560	
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1.อบรมเชิงปฏิบัติการการพัฒนาซอฟต์แวร์อิงแบบจำลอง							
2.วางแผนการดำเนินงาน - จัดหน้าที่เพื่อจัดทำในแต่ละหน้าที่ในการทำงาน - วางแผนระยะการทำงาน - กำหนดวันส่งงานความเคลื่อนไหวของงานในทุกๆสัปดาห์							
3.ติดตั้ง software - ติดตั้งโปรแกรม MATLAB &SIMULINK - ติดตั้ง support package ของ Freescale , raspberry pi							
4.ศึกษาข้อมูลเพิ่มเติมเกี่ยวกับอุปกรณ์และทดสอบการใช้งาน - ศึกษาการใช้งานและทดลองใช้กล้อง raspberry pi - ศึกษาการใช้งานบอร์ดและทดลองใช้ Freescale และ raspberry pi - ศึกษาการใช้งานและทดลองใช้ line scan camera - ทดสอบความเร็วมอเตอร์ที่ติดมากับรถ							
5. จัดหาและจัดซื้ออุปกรณ์นอกเหนือจากที่ได้รับมา - joystick - สาย mini USB - encoder							
6.ประชุมรายงานความคืบหน้า (ทุกๆสัปดาห์)							
7.ปรับปรุงแก้ไขระบบให้เกิดความเสถียรภาพมากขึ้น							
8.แจ้งปัญหาที่พบกับเจ้าหน้าที่ที่รับผิดชอบในการแข่งขัน - ปัญหาในส่วนของโปรแกรม - ปัญหาของระบบการเชื่อมต่อสื่อสารระหว่างอุปกรณ์							
9.ส่งรายงานประกอบการแข่งขัน (5 ก.พ. 2560)							
10.ทำการแข่งขัน (19 ก.พ. 2560)							
11.สรุปผลการแข่งขัน							

System Diagram



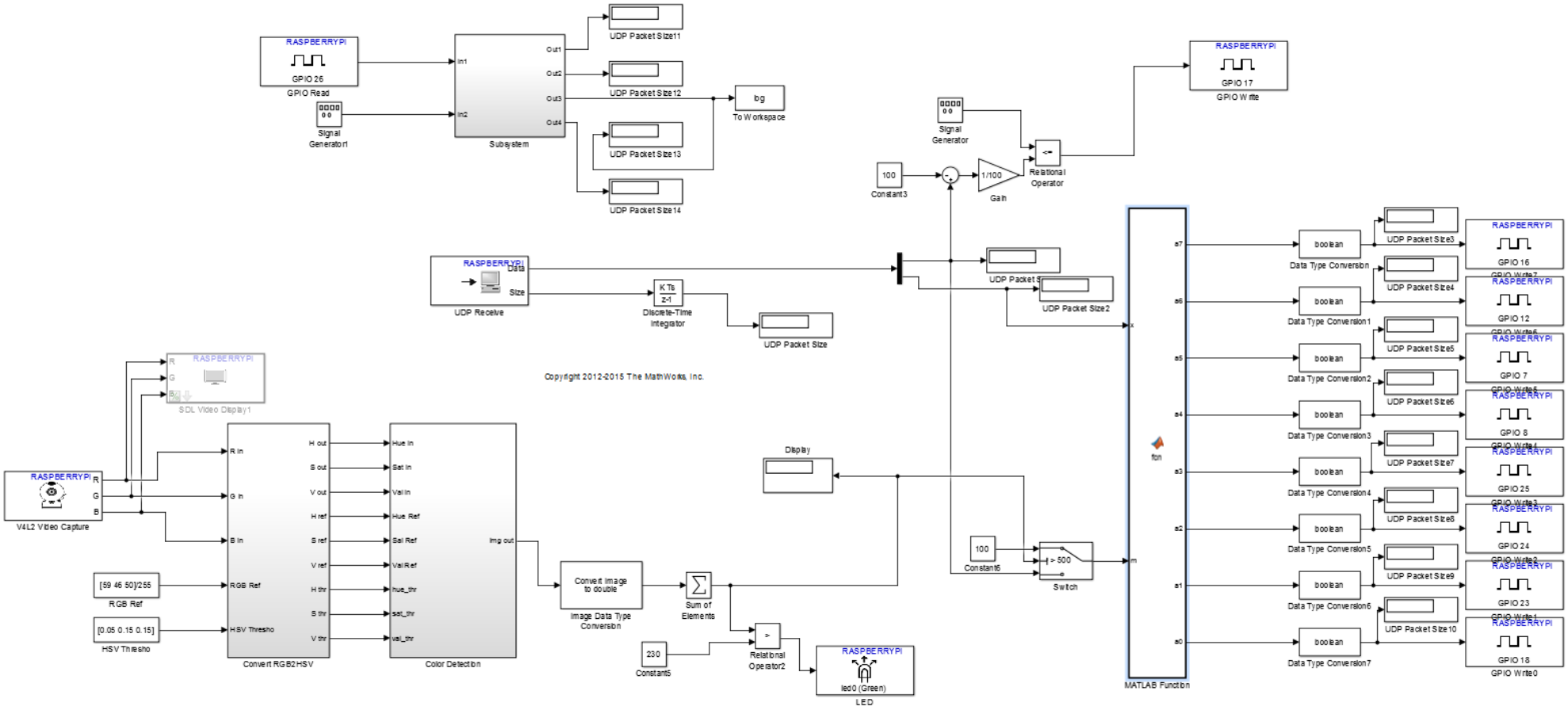
ใช้คอมพิวเตอร์เป็นสื่อกลางในการเชื่อมต่อระหว่าง Joystick และ Raspberry pi โดยคอมพิวเตอร์จะทำการรับค่าจาก Joystick ผ่านโปรแกรม Pycharm และแปลงเป็นการสื่อสารแบบ UDP ส่งไปยัง Raspberry pi โดย Raspberry pi จะทำการแปลงค่าต่างๆที่ได้รับมาเป็นข้อมูล 3 อย่างคือ

1. ความเร็วมอเตอร์
2. ทิศทางมอเตอร์
3. องศาของ Steering

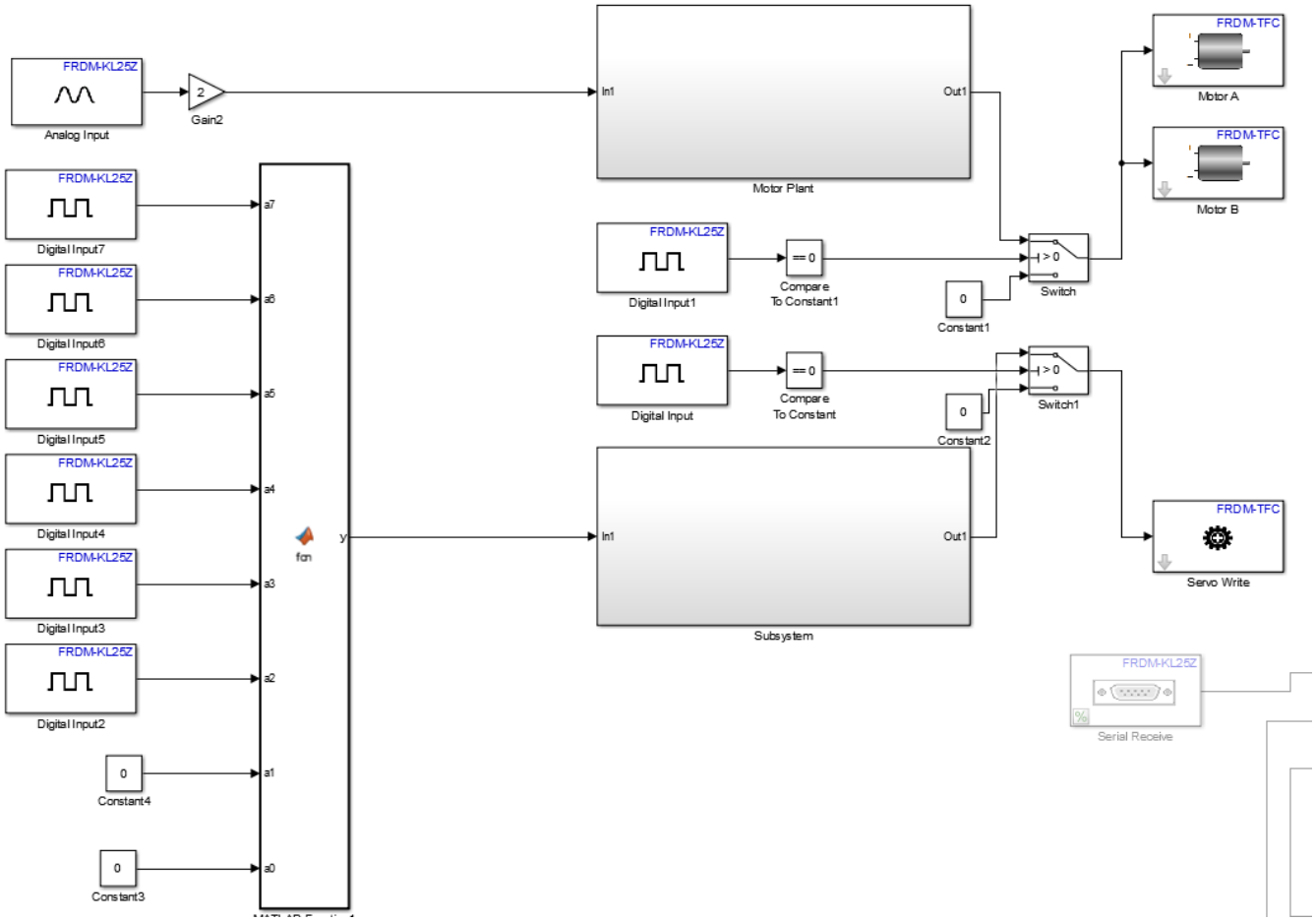
โดย Raspberry pi จะทำการส่งค่าต่างๆไปยัง Freescale board เพื่อทำการควบคุมมอเตอร์และเซอร์โวที่ใช้ขับเคลื่อน Steering โดยใช้สัญญาณ PWM ในการควบคุมความเร็วของมอเตอร์ และองศาของเซอร์โว โดยจะมี Encoder เชื่อมต่ออยู่กับ Raspberry pi คอยตรวจจับความเร็วและแสดงผลผ่านคอมพิวเตอร์

ในส่วนของกล้องจะเชื่อมต่อกับ Raspberry pi เพื่อตรวจจับไฟจราจร และทำการเปลี่ยนค่าที่ Raspberry pi จะส่งเพื่อไปสั่งการทำงานของมอเตอร์และเซอร์โว

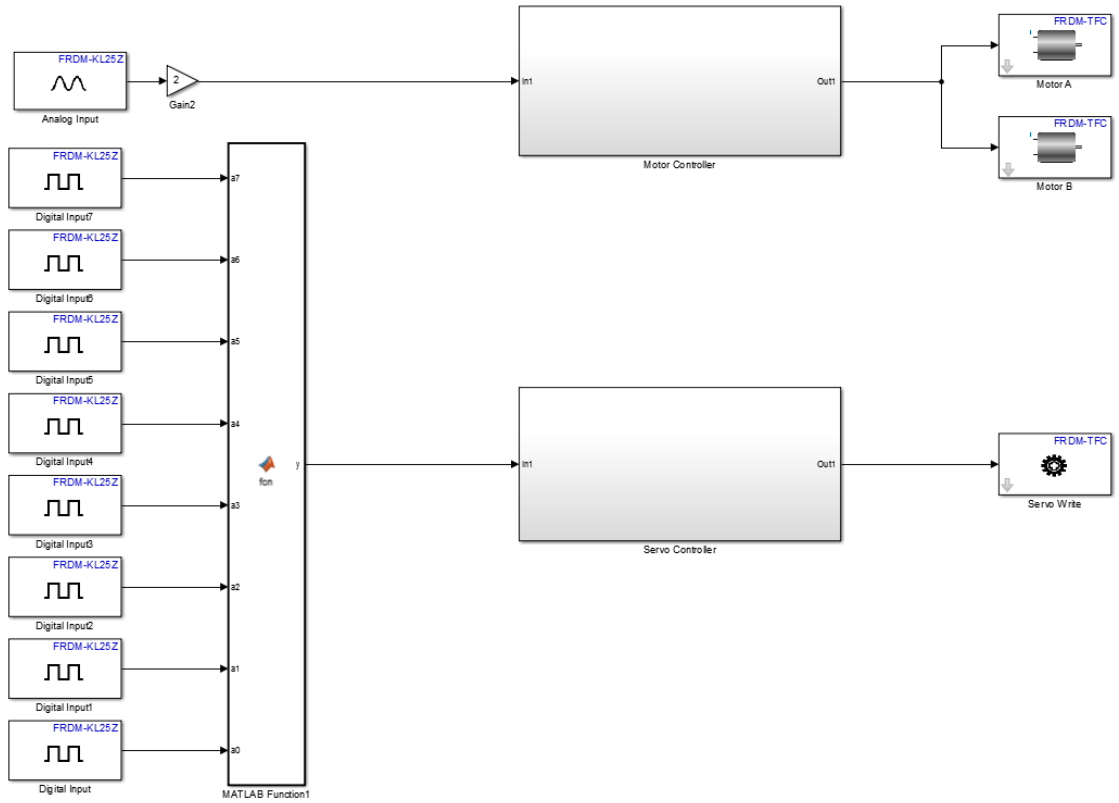
Simulink Model : Raspberry Pi



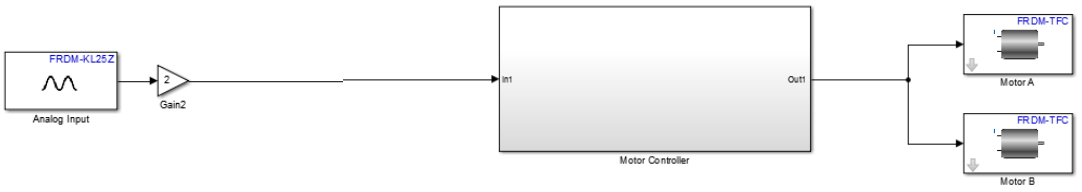
Simulink Model : freescale board



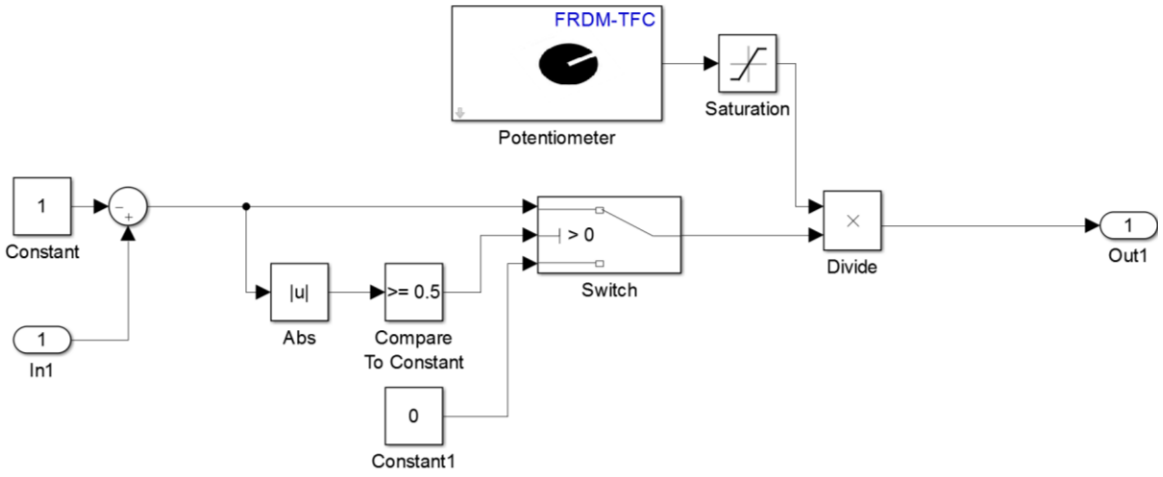
Simulink Model : Car controller on Freescale board



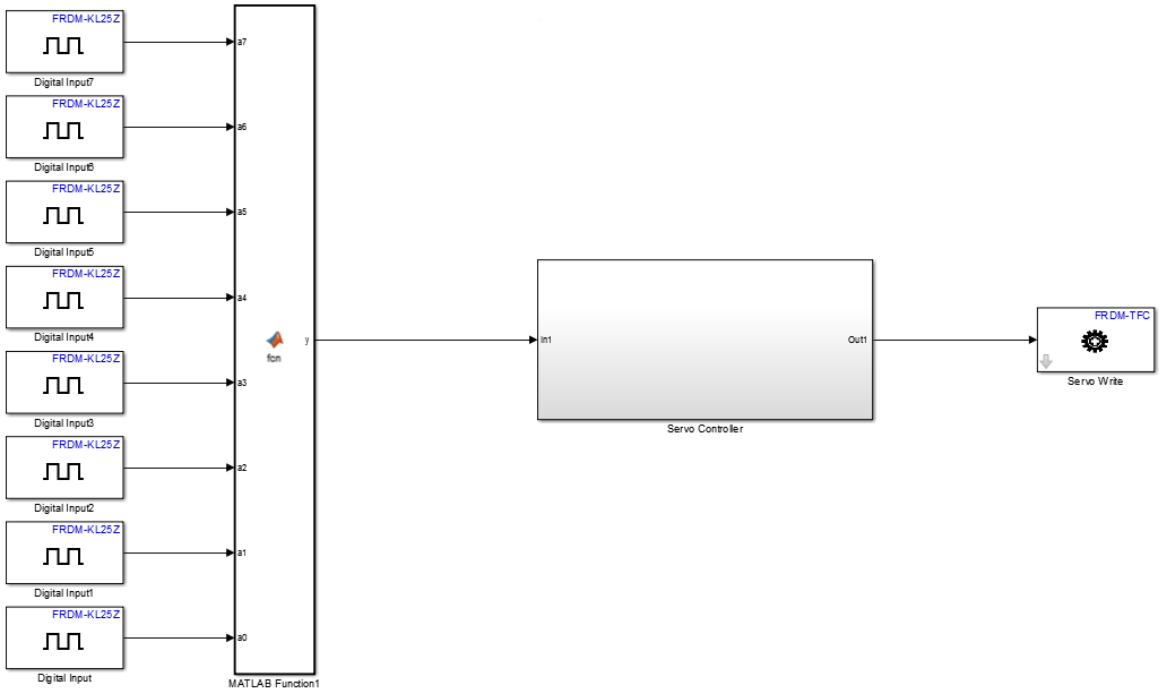
ในส่วนของการควบคุมกลไกต่างๆของรถนั้น จะรับข้อมูลจากคอนโทรลเลอร์ทั้งหมด 2 ส่วนและ 2 รูปแบบ ประกอบด้วย



1. การควบคุมมอเตอร์ ใช้การรับข้อมูลแบบ อนาล็อก (1 ตามรูป) โดยในฝั่งของ Raspberry pi จะทำการสร้างสัญญาณ อนาล็อก โดยใช้ PWM ซึ่งจะทำให้ได้แรงดันอยู่ในช่วง 0 – 3.3 โวลต์ และในฝั่งของการควบคุมรถ จะนำค่าที่อ่านได้อยู่ในช่วง 0.0 – 1.0 ไปคำนวณเพื่อขยายช่วงของค่าให้สามารถควบคุมมอเตอร์ได้ทั้ง 2 ทิศทาง โดยจะมีการลดสัญญาณรบกวนโดยใช้การเปรียบเทียบค่าที่ต่ำกว่าค่าหนึ่งที่กำหนดขึ้น เมื่อได้ค่าในการควบคุมมอเตอร์แล้วจึงนำมาคูณกับค่าของตัวต้านทานปรับค่าได้ที่อยู่กับ FRDM-TFC Shield เพื่อควบคุมความเร็วสูงสุดของการหมุนมอเตอร์



Motor Controller

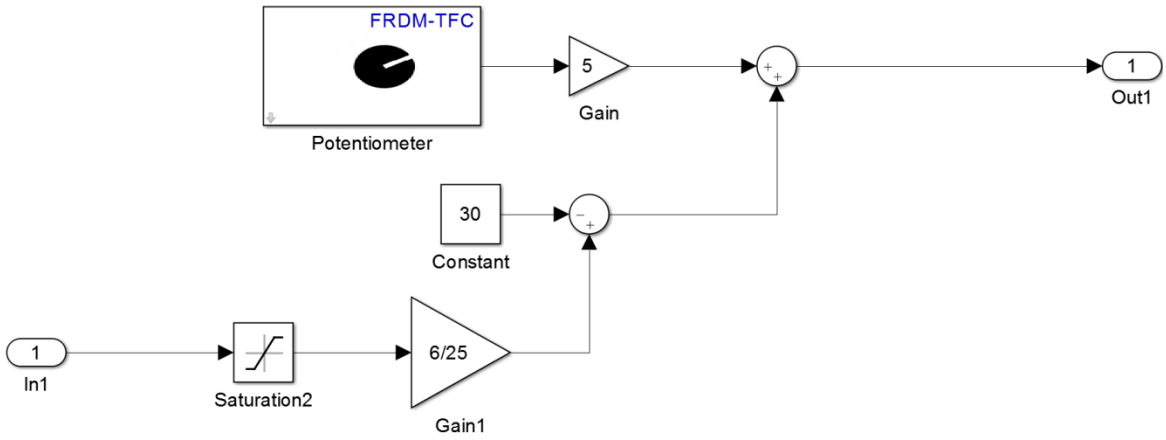


2. การควบคุมเซอร์โวมอเตอร์ จะใช้การส่งข้อมูลเป็นจำนวน 8 บิท โดยจะนำค่าในรูปของดิจิตอลที่ได้จากพินทั้ง 8 พิน นำมาคำนวณเป็นค่าที่อยู่ในช่วง 0 – 255 และนำค่าที่ได้ทำการคำนวณเปลี่ยนช่วงเพื่อนำไปใช้ในการควบคุมเซอร์โวมอเตอร์ที่มีค่าอยู่ในช่วง -30 – 30 โดยจะมีการใช้ตัวต้านทานปรับค่าได้เพื่อปรับองศาเริ่มต้น (องศาที่ 0 ตามคำสั่งจาก Raspberry pi) ให้รถมีองศาเริ่มต้นที่ตรงให้ง่ายต่อการควบคุม

```
function  y = fcn(a7,a6,a5,a4,a3,a2,a1,a0)
%#codegen

y = (a7*2^7)+(a6*2^6)+(a5*2^5)+(a4*2^4)+(a3*2^3)+(a2*2^2)+(a1*2^1)+(a0*2^0);
```

ฟังก์ชั่นการคำนวณค่า 8 พิน ในอยู่ในรูปของข้อมูล 8 บิท

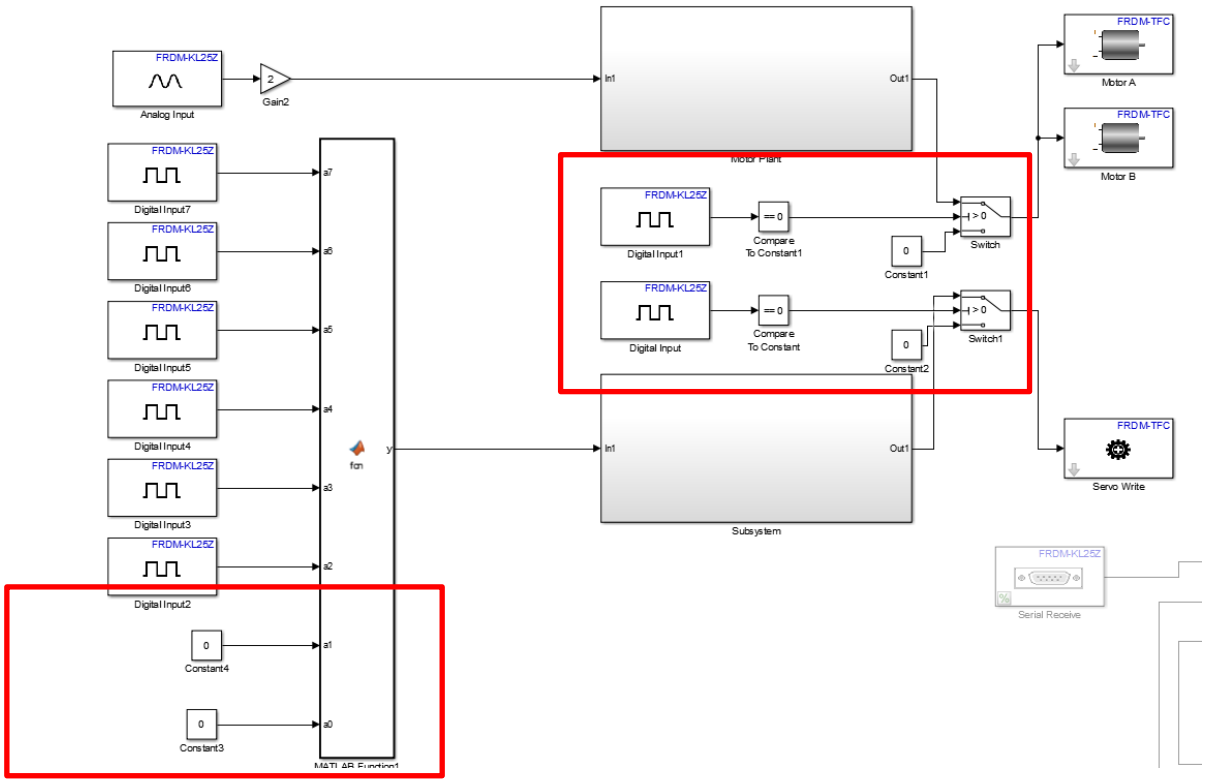


Servo Controller

ปัญหาที่พบ

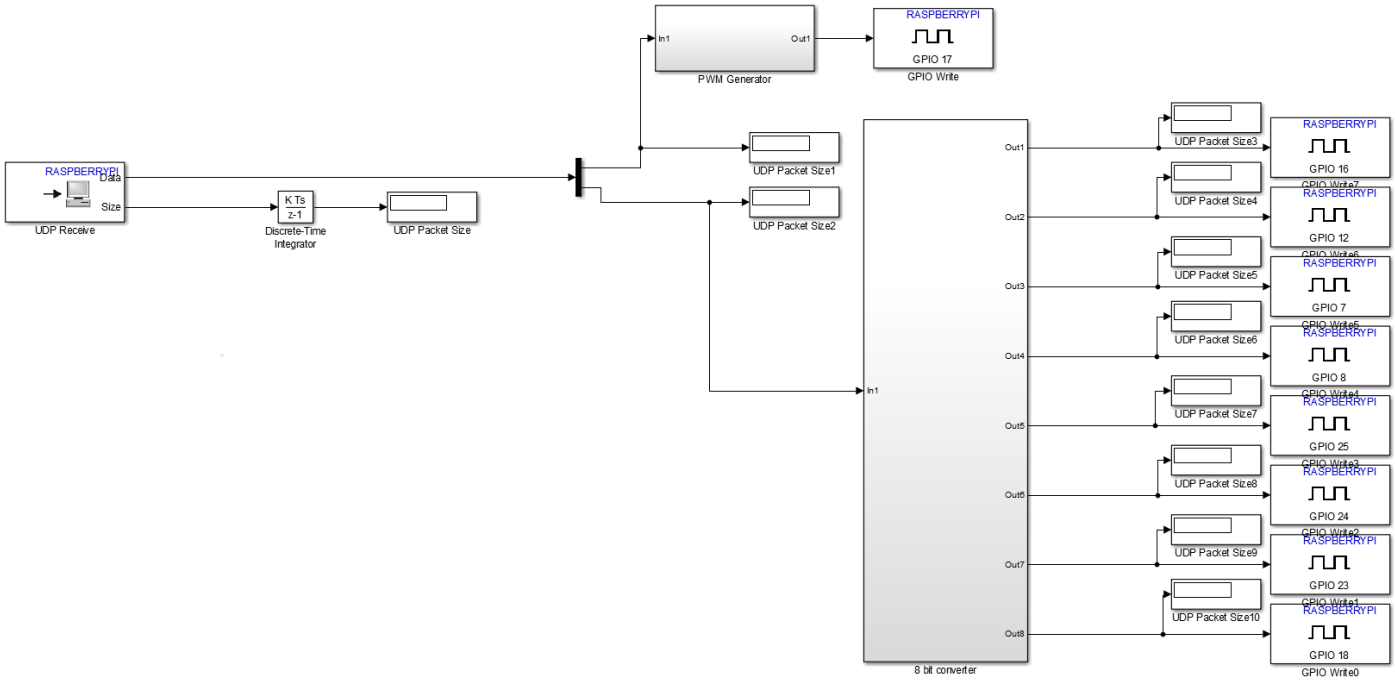
จากโมเดลทั้งหมดข้างต้นนั้น พบปัญหาอยู่ 2 อย่างดังนี้

1. เนื่องจากบอร์ดคอนโทรลเลอร์และมอเตอร์สามารถตอบสนองต่อความถี่สูงได้ ทำให้การส่งความถี่จาก Raspberry pi ส่งผลให้มอเตอร์มีการทำงานตลอดเวลา ทำให้มอเตอร์ร้อน และมีโอกาสที่จะหมุนไปทิศทางใดทางหนึ่งในขณะที่ส่งค่าความเร็วเป็น 0
2. เมื่อสัญญาณของรถขาดการเชื่อมต่อ จากโมเดลและรูปแบบการสื่อสารข้างต้นทำให้ในสถานะปกติ รถจะถอยหลัง และ Steering ด้านหน้าจะหันไปทางขวาตลอดเวลา ทำให้อาจเกิดอุบัติเหตุขึ้นได้ระหว่างการขับขี่ จึงได้เพิ่มเติมในส่วนของการตรวจสอบการเชื่อมต่อเข้าไปเพื่อเป็นการเปิดหรือปิดการทำงานของรถโดยอัตโนมัติ



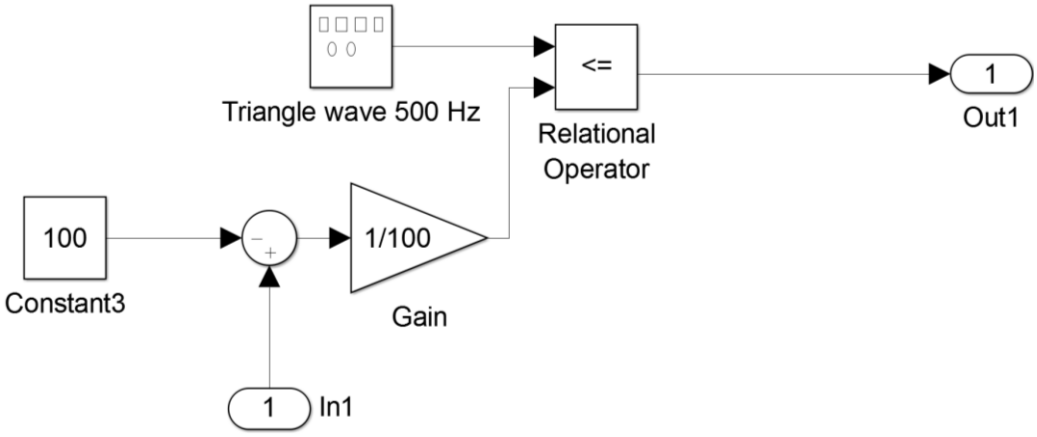
โดยทำการเปลี่ยนแปลงสังเกตได้จากรูป (กรอบสี่เหลี่ยมสีแดง) โดยการนำพินที่ใช้รับข้อมูลบิทต่ำซึ่งมีผลต่อการควบคุมน้อย มาใช้งาน โดยการตรวจสอบการดึงสัญญาณเป็น Low ของ Raspberry pi(Default pin's logic : high) โดยหากไม่เกิดการดึงสัญญาณจากฝั่ง Raspberry pi ระบบจะทำการส่งค่า 0 ไปสั่งการมอเตอร์และเซอร์โวให้อยู่ในท่าเริ่มต้นของรถยนต์

Simulink Model : Communication Part of Raspberry pi



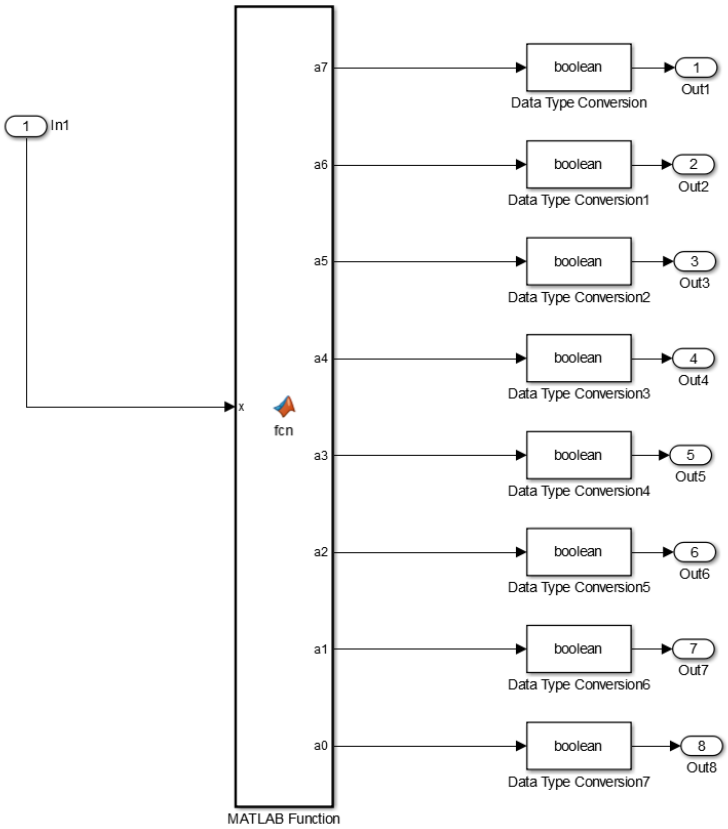
ในส่วนของการสื่อสารกันนั้น จะทำการรับค่าจาก joystick ผ่านการเชื่อมต่อไร้สายในระบบ Wifi โดยใช้การสื่อสารแบบ UDP เมื่อรับค่ามาได้ก็จะนำไปรวมกับระบบการประมวลผลภาพเพื่อเปลี่ยนแปลงค่าในกรณีที่ตรวจพบไฟแดง หรือระบบอัตโนมัติช่วยเหลือคนขับในการควบคุมเลน โดยเมื่อได้ค่ามาแล้วจึงนำไปทำการคำนวณเพื่อแปลงให้อยู่ในรูปของการส่งสัญญาณข้อมูล โดยจะแบ่งออกเป็น 2 ส่วน ประกอบด้วย

1. ส่วนของการควบคุมมอเตอร์ ในส่วนนี้ระบบจะทำการสร้างสัญญาณ Triangle wave ขึ้นมา โดยการกำหนด ค่าสูงสุดและต่ำสุดของสัญญาณให้อยู่ในช่วงของค่าที่ได้รับมา และนำค่าที่ได้รับมาทำการเปรียบเทียบกับสัญญาณที่สร้างขึ้นเพื่อสร้าง PWM แล้วทำการส่งออกทางพิน GPIO เพื่อนำไปใช้ในการควบคุมความเร็วของมอเตอร์ต่อไป



PWM Generator

2. ส่วนของการควบคุมองศาของเซอร์โว ในส่วนนี้จะนำค่าที่ได้รับจากการส่งข้อมูลจาก Joystick และการคำนวณจากระบบควบคุมเลนแล้ว จะนำค่าที่ได้มาทำการแปลงให้อยู่ในรูปข้อมูล 8 บิท และนำไปสั่งการเปิดปิดพินต่างๆ ของ Raspberry pi เพื่อทำการส่งข้อมูลไปยังส่วนของ ระบบควบคุมรถต่อไป

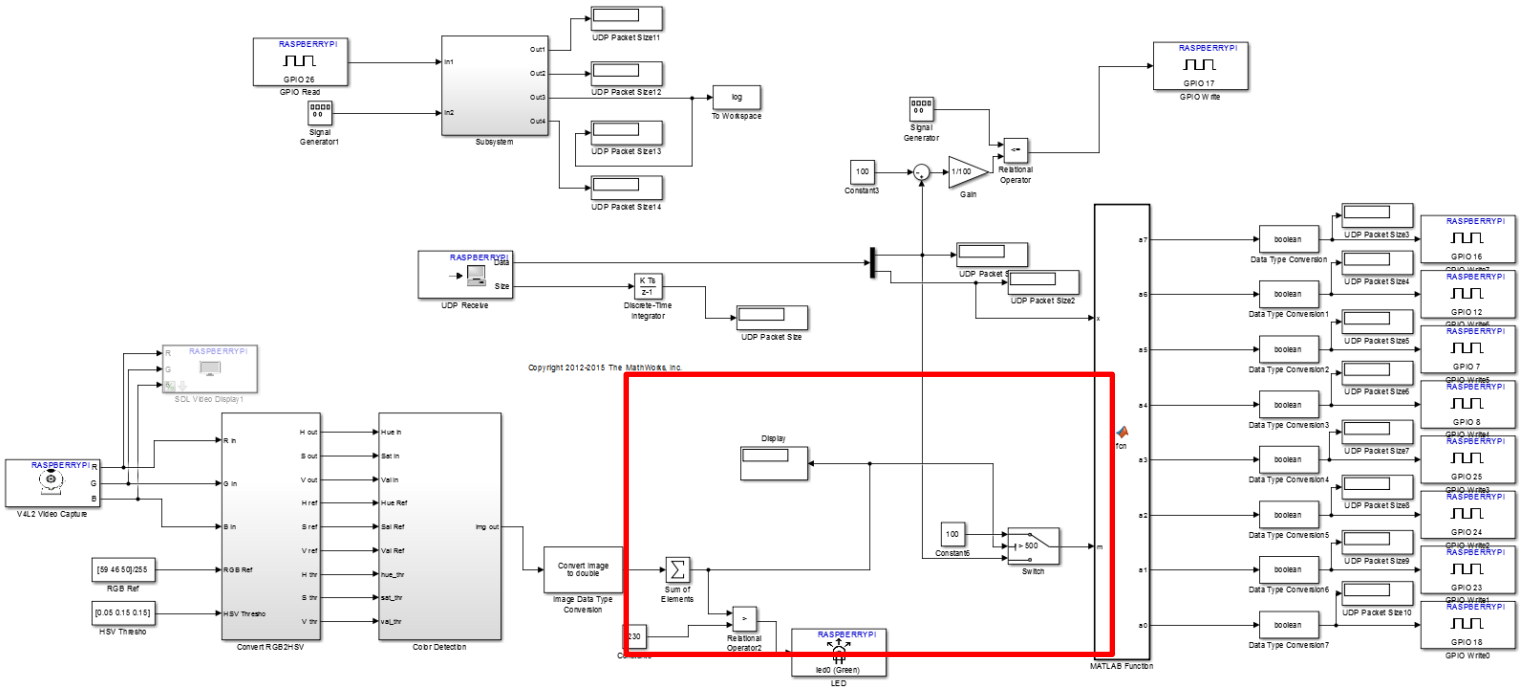


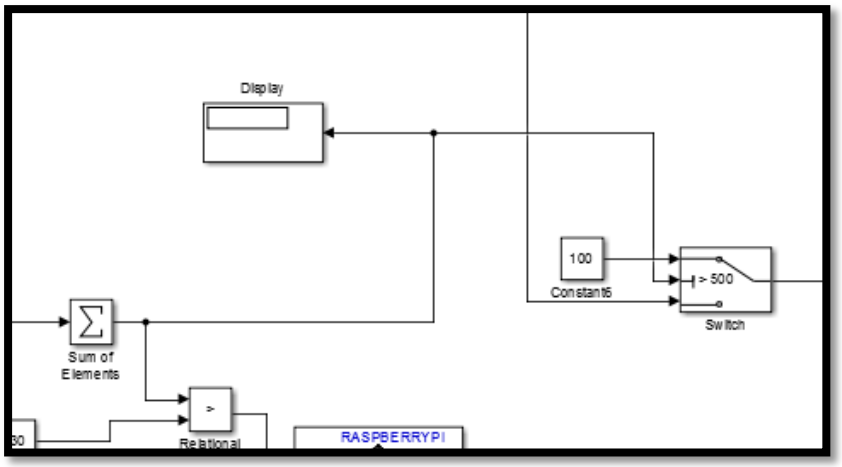
8 Bits Converter

```
function [a7,a6,a5,a4,a3,a2,a1,a0]= fcn(x)
%#codegen
y = [0 0 0 0 0 0 0 0];
u = x;
if(u > 127)
    y(8) = 1;
    u = u - (128);
end
if(u > 63)
    y(7) = 1;
    u = u - (64);
end
if(u > 31)
    y(6) = 1;
    u = u - (32);
end
if(u > 15)
    y(5) = 1;
    u = u - (16);
end
if(u > 7)
    y(4) = 1;
    u = u - (8);
end
if(u > 3)
    y(3) = 1;
    u = u - (4);
end
if(u > 1)
    y(2) = 1;
    u = u - (2);
end
if(u > 0)
    y(1) = 1;
end
[a0,a1,a2,a3,a4,a5,a6,a7] = y;
```

8 Bits Converter Code

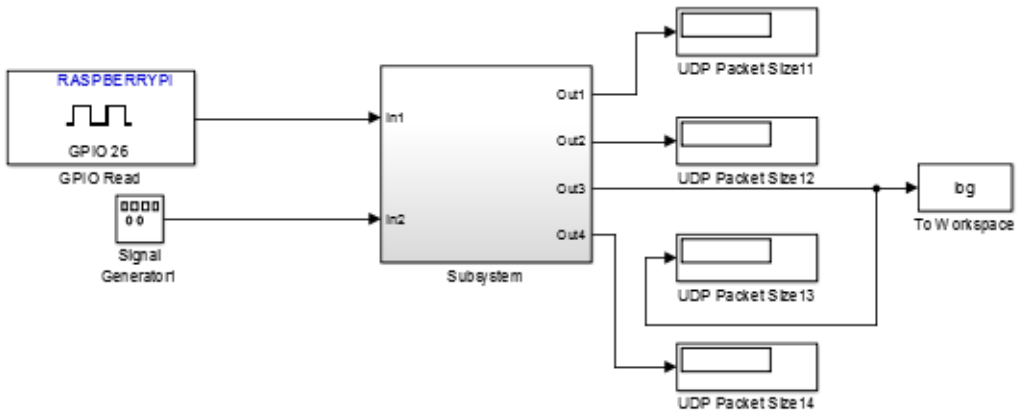
โดยจากประเด็นปัญหาจากการควบคุมรถยนต์ตามที่กล่าวไว้ในรายงาน Model ในส่วนของคอนโทรลเลอร์รถยนต์และในส่วนของการตรวจจับสัญญาณไฟ ทำให้เพิ่มเติมขึ้นมาในส่วนของการตัดการทำงานของคันเร่ง และการ acknowledge ของบอร์ด Raspberry pi โดยการดึงสัญญาณเป็น Low



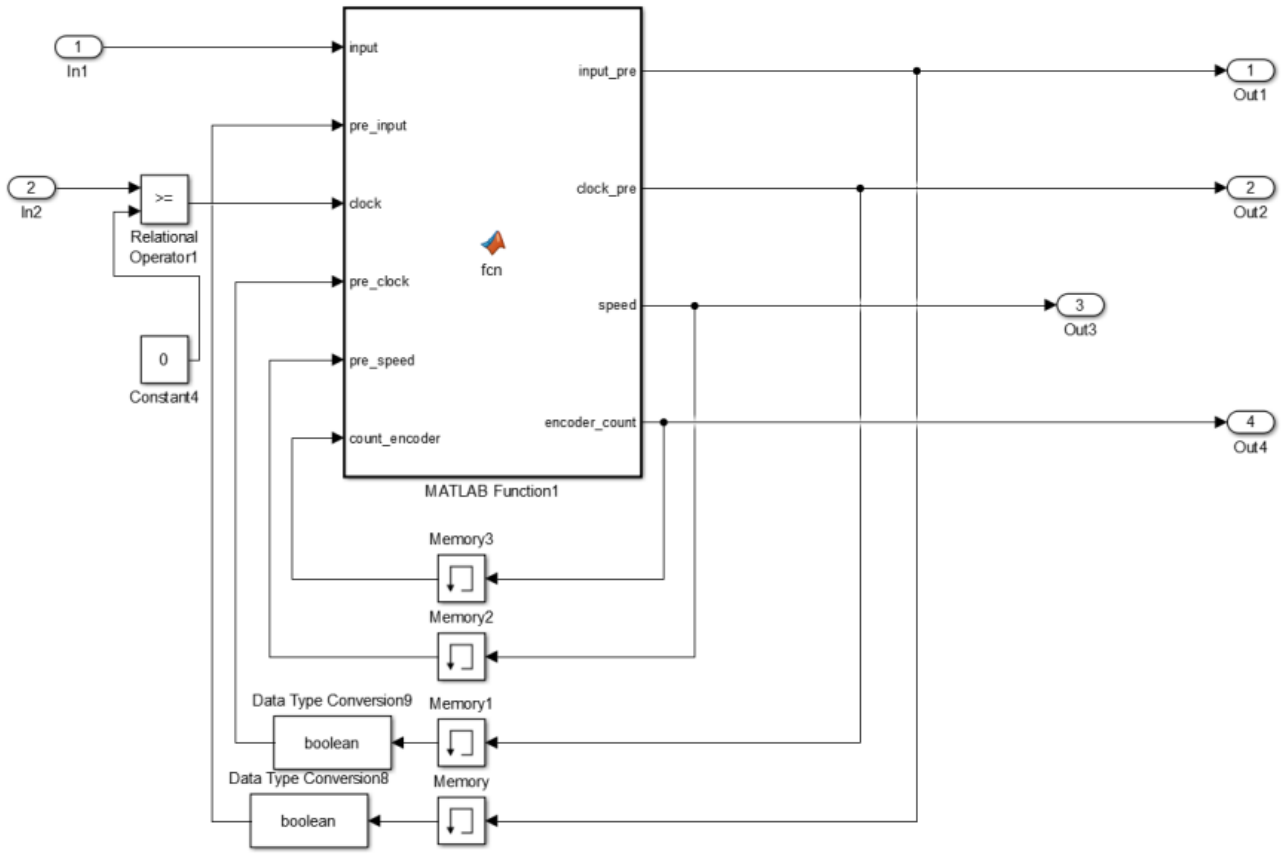


โดยภายใน MATLAB Function จะทำการตรวจสอบค่า 100 (รถหยุดนิ่ง) แล้วนำไปดึง High ที่พิน acknowledge ให้รถทำการหยุดโดยอัตโนมัติ ซึ่งค่าที่จะนำไปตรวจสอบหรือสั่งการมอเตอร์จะถูกควบคุมโดยจำนวน pixel ของสีที่เราต้องการตรวจสอบ (ณ ที่นี้คือสัญญาณไฟจราจร) ซึ่งจะนำจำนวน pixel มาทำการตรวจสอบและตัดการทำงานของคันเร่งโดยการใส่ค่า 100 เข้าไปในระบบเพื่อให้รถหยุด

Simulink Model : Encoder in Raspberry pi



ใน Model นี้จะรับสัญญาณ pulse จาก encoder เข้ามาและนำมาคำนวณเป็นความเร็วและสร้าง log file เพื่อตรวจสอบและสังเกตการทำงานของรถ โดยจะสร้างสัญญาณ square wave ขึ้นมาเพื่อคอย sampling ค่าต่างๆ ซึ่งหลักการทำงานคือเมื่อถึงจังหวะการ sampling ของ square wave ระบบจะทำการคำนวณค่า encoder ที่นับได้และทำการแปลงเป็นความเร็วในหน่วย rpm



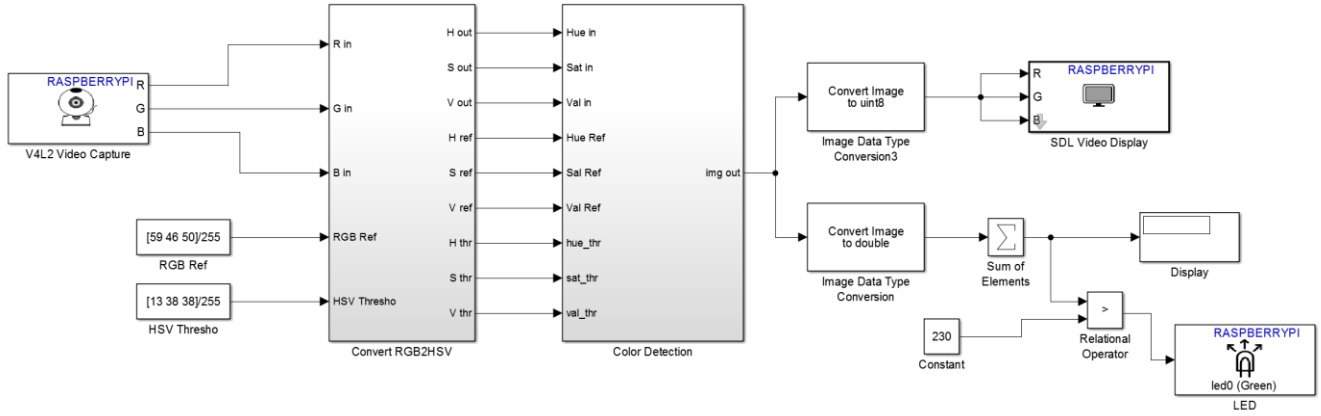
Encoder Model

```
function [input_pre,clock_pre,speed,encoder_count]= fcn(input,pre_input,clock,pre_clock,pre_speed,count_encoder)
input_pre = input ;
clock_pre = clock;
speed = pre_speed;
encoder_count = count_encoder;
if(input==1 && pre_input==0)
    encoder_count = encoder_count+1;
end
if(clock==1 && pre_clock==0)
    speed = (encoder_count*120)/6;
    encoder_count = 0;
end

%#codegen
```

Encoder Code

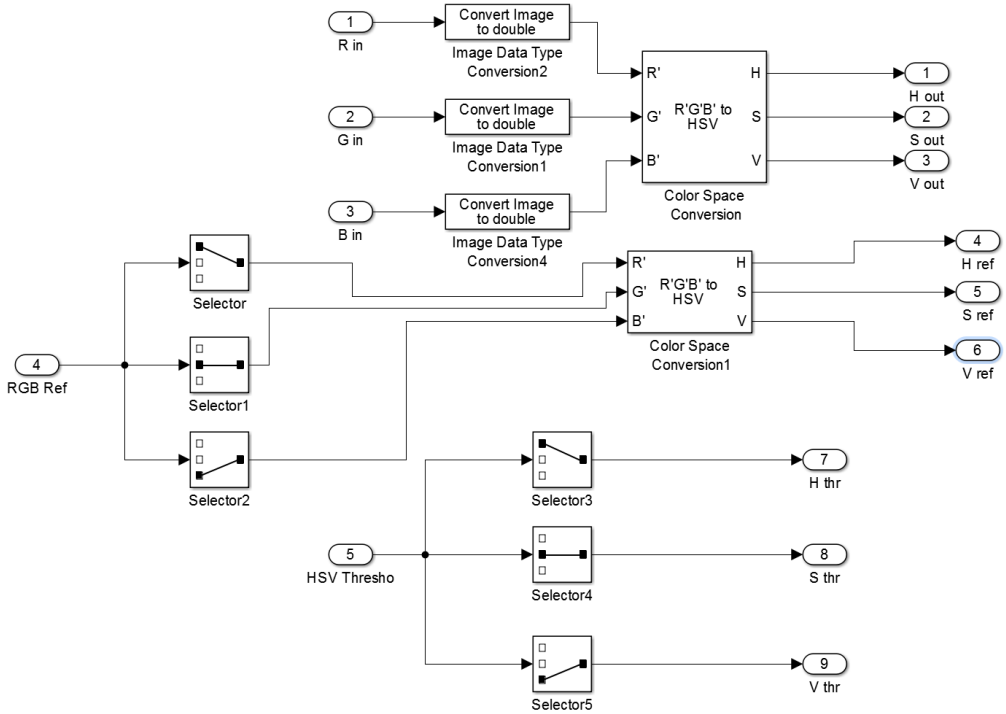
Simulink Model : Traffic Light Detect



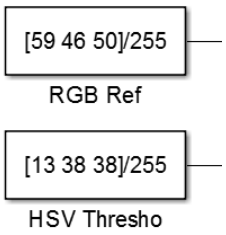
เป็นส่วนของการตรวจจับสีและขนาดของไฟ LED เพื่อที่จะสั่งให้รถสามารถหยุดได้ก่อนถึงเส้นสีขาว เมื่อตรวจพบเจอไฟจราจรสีแดง

ประกอบไปด้วย

CONVERT RGB2HSV BLOCK



เป็นการแปลง Color space จาก RGB เป็น HSV เพื่อให้ง่ายต่อการเช็คค่าความเข้มและความสว่างของสี และแปลงจาก unit8 ของ Camera module เป็น double เพื่อที่จะให้สามารถเข้า Block Color Space Conversion ได้

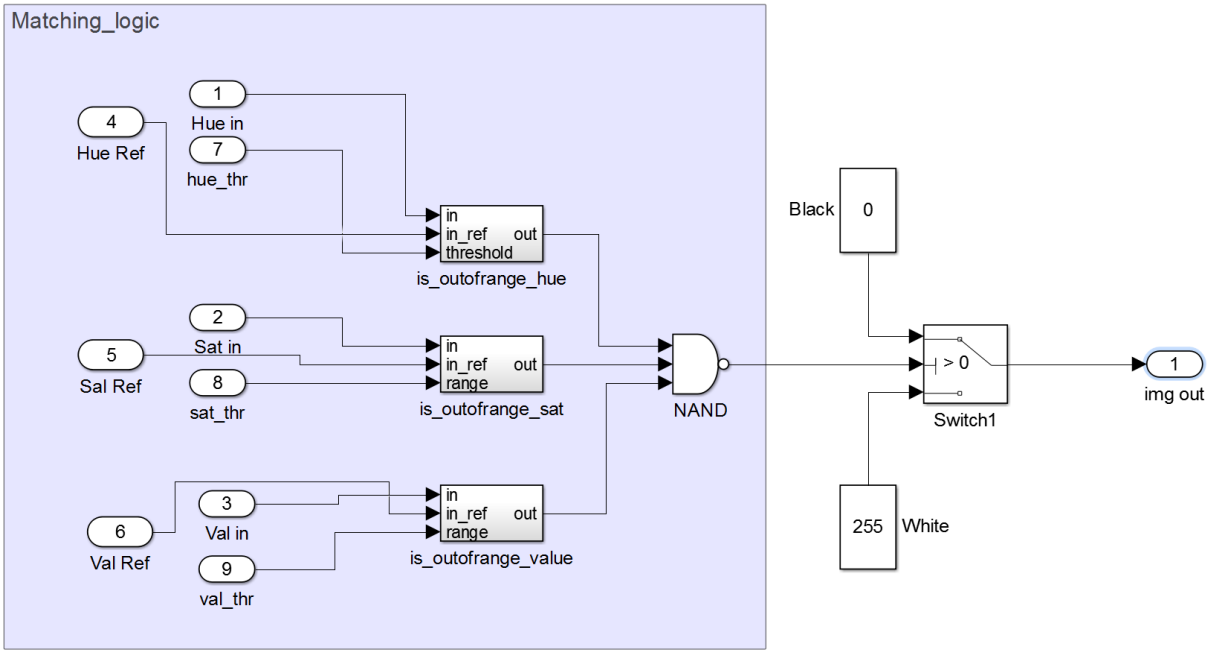


Input เป็น สัญญาณ R,G,B จาก Camera Module ,

- RGB Reference ใช้ตรวจจับสีของไฟจราจร
- HSV Threshold ใช้ในการปรับช่วงของสีที่สามารถตรวจจับได้

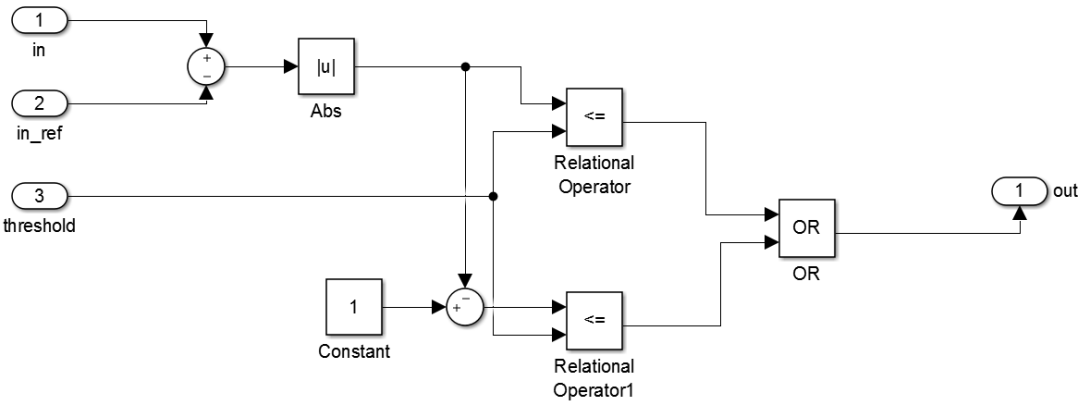
หมายเหตุ : ที่ต้องหารด้วย 255 เพราะเป็นการปรับค่าให้เป็นทศนิยม เนื่องจากสัญญาณที่เข้าบล็อก uint8 to double จะกลายเป็นทศนิยมช่วง 0-1

COLOR DETECTION BLOCK

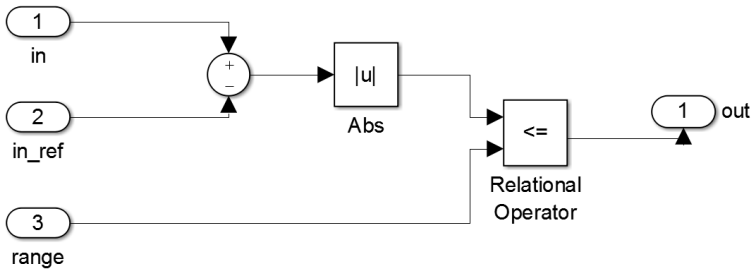


มีหน้าที่เช็คว่าสัญญาณที่เข้ามาอยู่ในช่วงที่ต้องการหรือไม่ โดยจะเช็คว่าสัญญาณนั้นต้องอยู่ใน range ที่กำหนดทั้ง 3 Channel HSV โดยที่บล็อกนี้จะมี Output เป็นสัญญาณรูปภาพขาวดำ ตามแต่ละจุดที่เช็คได้ หากจุดไหนเป็นสีที่ต้องการ จุดนั้นก็จะมี Output เป็นสีขาว หากไม่ใช่ก็จะเป็นสีดำ

is_outofrange_hue BLOCK

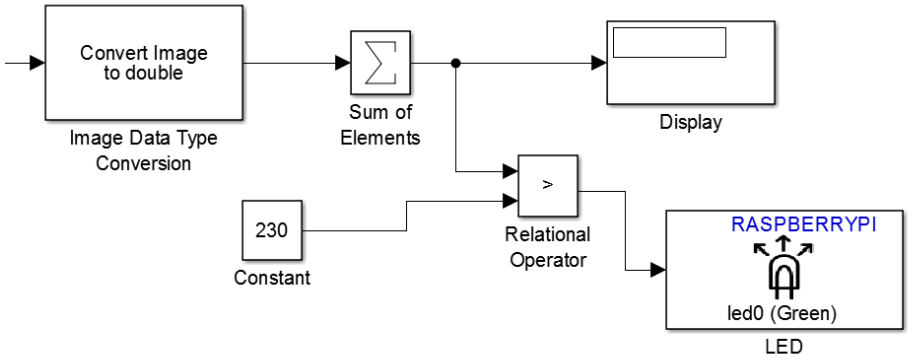


is_outofrange_sat BLOCK & is_outofrange_val BLOCK



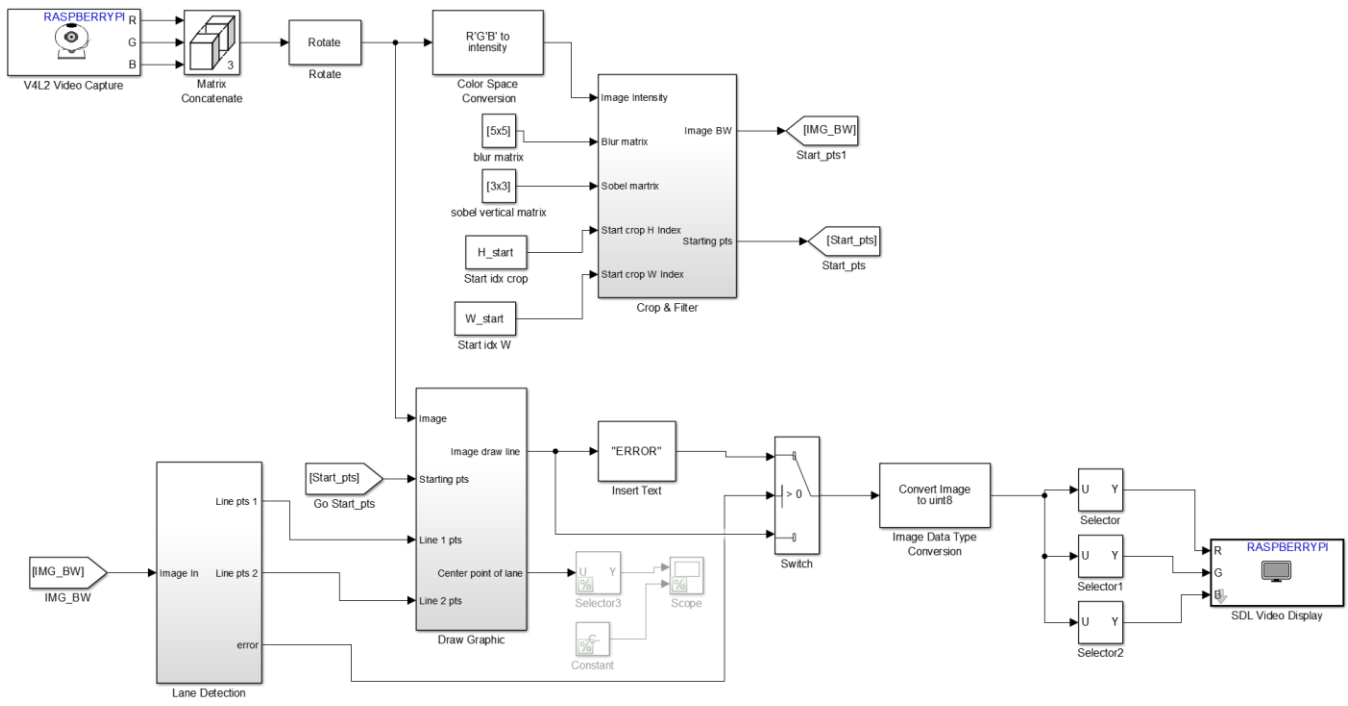
เป็นบล็อกที่ใช้เช็คค่าสีว่าอยู่ในช่วงที่กำหนดหรือไม่ โดยมี Output เป็น Boolean

การตรวจจับขนาดของสัญญาณไฟจราจร



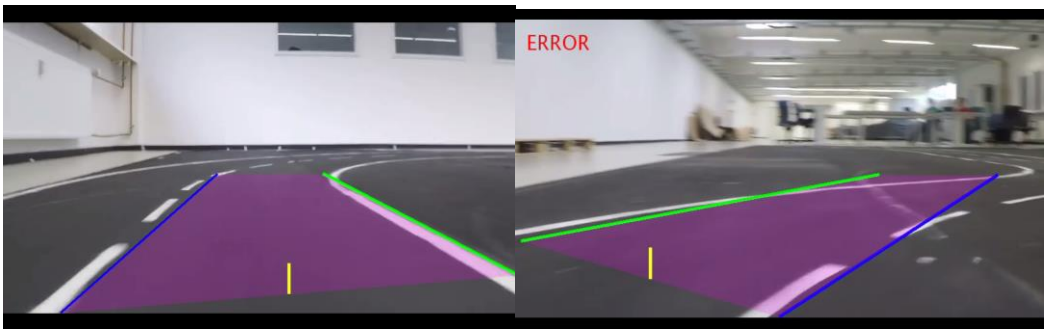
ใช้ผลรวมของค่าในทุกๆ Pixel ของรูป เราก็จะได้จำนวนจุดที่มีสีตรงตามที่ต้องการ ยิ่งมีจำนวน Pixel เยอะ แสดงให้เห็นว่ารถของเรายิ่งเข้าใกล้สัญญาณไฟจราจร

Simulink Model : LANE TRACKING MODEL (EXTRA Model)



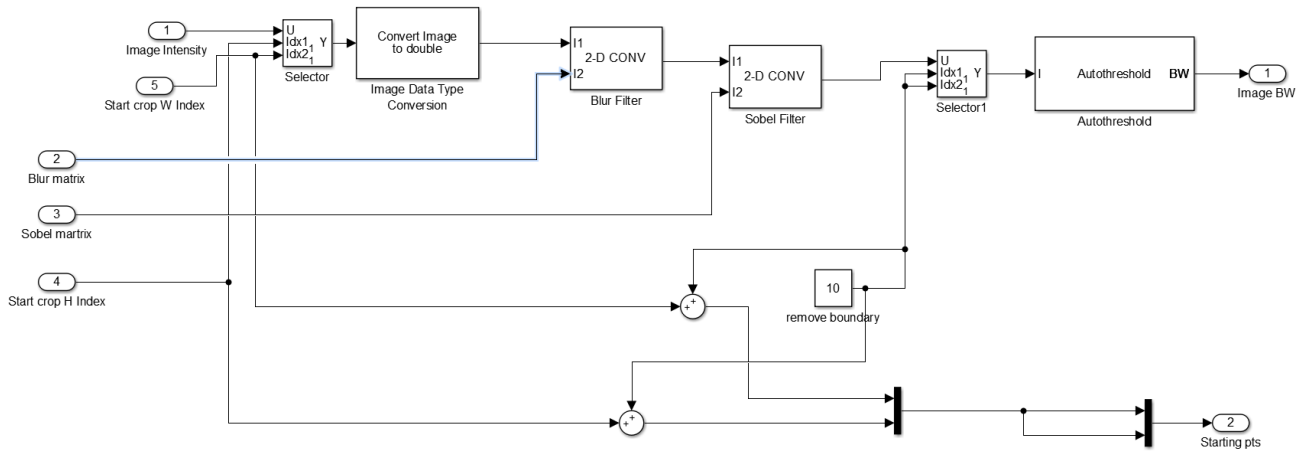
เป็นโมเดลที่ทำให้รถสามารถวิ่งไปตามเลนได้เองโดยไม่ต้องมีคนบังคับ โดยมี Input เป็นสัญญาณรูปภาพจาก Camera Module และ ค่า Parameter ในการปรับรูปภาพให้เหมาะสมกับการ tracking (เช่น blur matrix , Field of interest)

มี Output เป็นจุดกลางของเลน (ใช้ในการเป็นจุดอ้างอิงในการปรับองศาล้อหน้า ให้รักษาจุดกึ่งกลางของรถให้ตรงกับจุดกึ่งกลางของเลน) , สัญญาณ error เมื่อไม่สามารถตรวจจับเลนได้ หรือเลนมีลักษณะผิดปกติ (เพื่อตัดการวิ่งอัตโนมัติ และเปลี่ยนไปใช้คนบังคับแทน) , สัญญาณรูปภาพ แสดงลักษณะของเลนที่ตรวจจับได้



หมายเหตุ : ในรูปเป็นการนำคลิปวิดีโอเลนถนนจำลองใน Youtube Channel ของ Clemens Elflein มาทดสอบกับโมเดล Lane Tracking

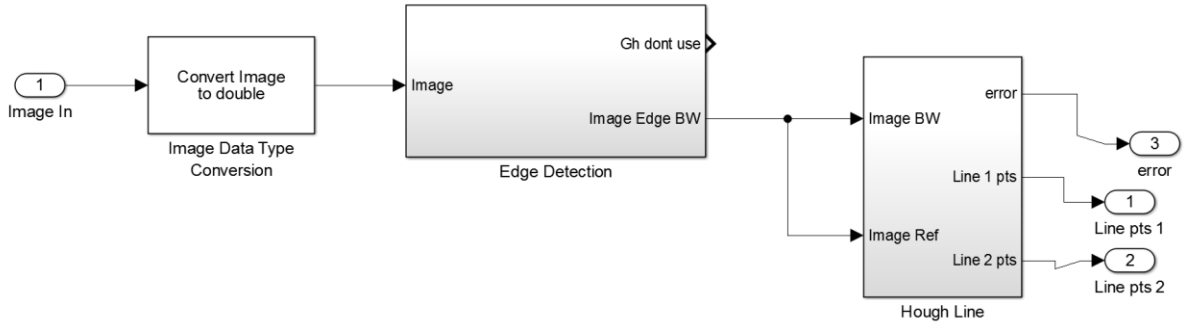
Crop & Filter BLOCK



เป็นการ crop รูปเพื่อตัดส่วนที่ไม่จำเป็นในการ Tracking lane และนำส่วนนั้นไป Blur เพื่อลด Noise ออกจากนั้นนำไปหา Edge Detection โดยใช้ Sobel vertical เนื่องจากเลนที่กล้องจับได้จะมีลักษณะอยู่ในแนว vertical

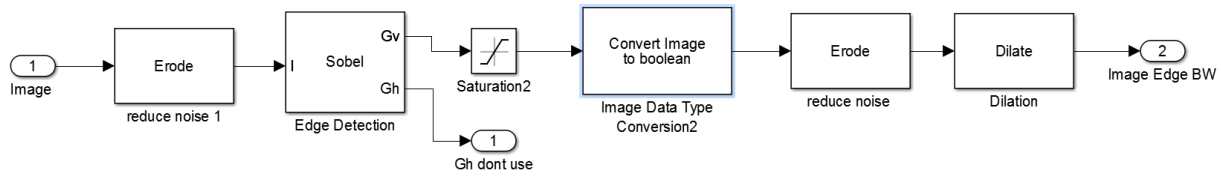
มี Output เป็นสัญญาณรูปขาวดำ และจุดเริ่มต้นของการ crop รูปภาพ เพื่อใช้อ้างอิงเปรียบเทียบจากรูปเต็ม

Lane Detection BLOCK



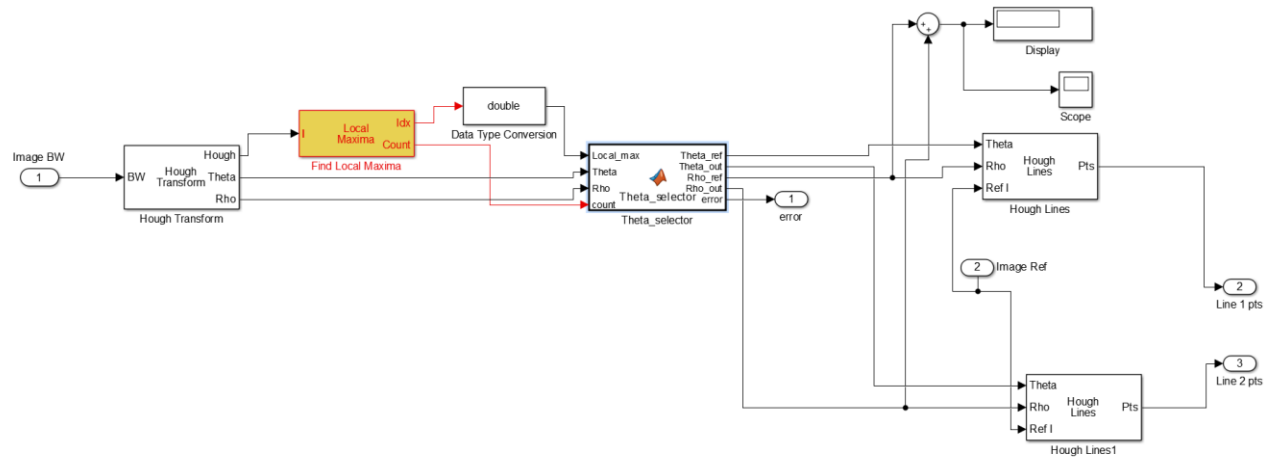
เป็นส่วนในการ detect ว่าเส้นของ lane อยู่ที่จุดในของรูปภาพ ประกอบไปด้วย

Edge Detection BLOCK



เป็นการลด Noise ประเภท Salt,pepper และทำ Edge Detection เพื่อเพิ่มความชัดเจนของเลน

Hough Line BLOCK



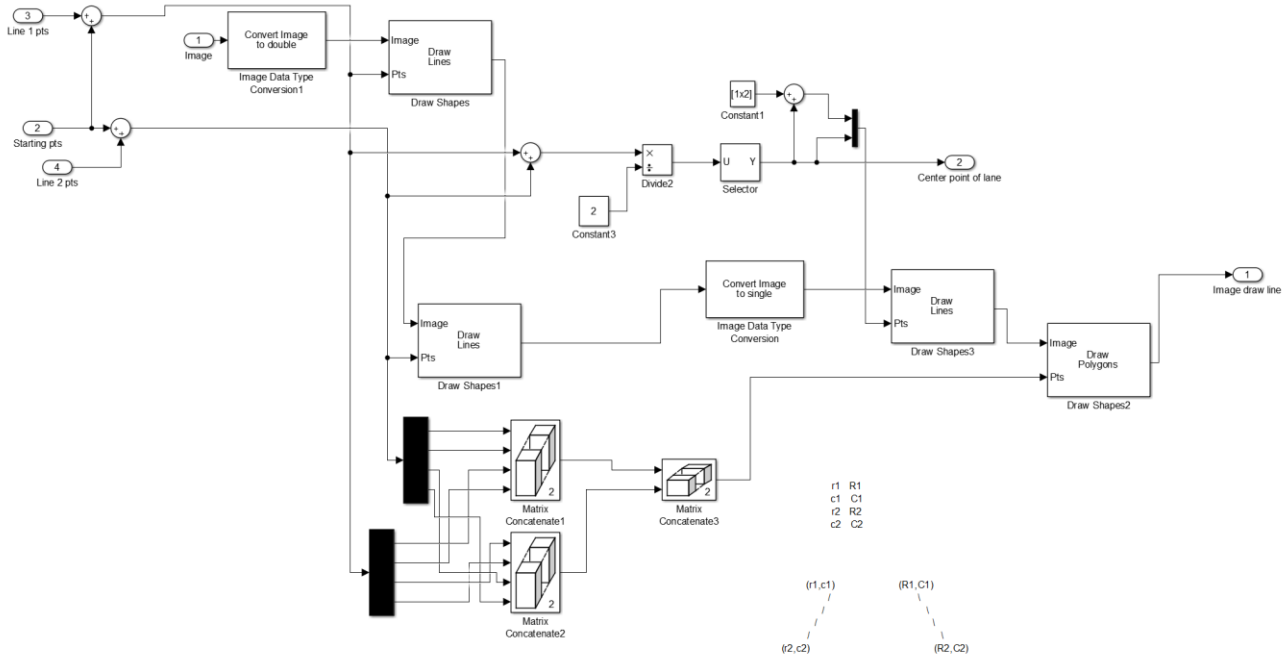
เป็นการทำ Hough Tranform เพื่อที่จุดหาเส้นต่างๆที่อยู่ในรูปภาพ และเลือกเส้นที่ชัดเจนที่สุด โดยจะมี Theta_selector เป็นการเลือกองศาของเส้นที่จะสามารถเป็นเลนได้ และมี Output เป็นตำแหน่งของเส้นเลน 2 เส้น , และสถานะของ Error

Theta_Selection FUNCTION

ปัญหาที่ต้องใช้ฟังก์ชั่นนี้ คือ เส้นที่ชัดที่สุดจาก Hough line 2 เส้นอาจจะเป็นเส้นที่ซ้อนทับกัน อยู่ในตำแหน่งและมีองศาใกล้ๆกัน มันจึงไม่ใช่ Lane ที่ต้องการ ฟังก์ชั่นนี้จะมีวิธีการ คือ เลือกเส้นที่ชัดเจนที่สุดของ Hough line ไว้ใช้อ้างอิง จากนั้นหาเส้นที่ชัดเจนที่สุดที่มีองศาของเส้นนั้นบวกกับองศาของเส้นอ้างอิงแล้วเข้าใกล้ 0 มากที่สุด (เช่น เส้นอ้างอิงเป็น 30 องศา เส้นอีกฝั่งของเลนควรจะประมาณ -30 องศา เมื่อนำมาบวกกันจะได้ประมาณ 0) และถ้าหากหาไม่เจอ จะส่ง error ไป

```
function [Theta_ref,Theta_out,Rho_ref,Rho_out,error] = Theta_selector(Local_max,Theta,Rho,count)
[H,W] = size(Local_max);  error = 0;  mem = 100;  idx_out = 1;
if count > 1
for x = 1:H
   if Theta(Local_max(x,1)) < 0
      Theta_ref = Theta(Local_max(x,1));
      Rho_ref = Rho(Local_max(x,2));
      break;
   else
   Theta_ref=Theta(Local_max(x,1));
   Rho_ref = Rho(Local_max(x,2));
   end
end
%% abs
for x = 1:H
A = abs(Theta(Local_max(x,1)) + Theta_ref);
   if A < mem
      mem = A; idx_out = x;
   end
end
Theta_out = Theta(Local_max(idx_out,1));
Rho_out = Rho(Local_max(idx_out,2));
%% error
if Theta_out > 1 | Theta_ref > 1
      error = 1;
end
if (Theta_out > 0 && Theta_ref > 0) | (Theta_out < 0 && Theta_ref < 0)
      error = 1;
end
else
   Theta_ref=Theta(Local_max(1,1));
   Rho_ref = Rho(Local_max(1,2));
   Theta_out=Theta(Local_max(1,1));
   Rho_out = Rho(Local_max(1,2));
end
```

Draw Graphic BLOCK



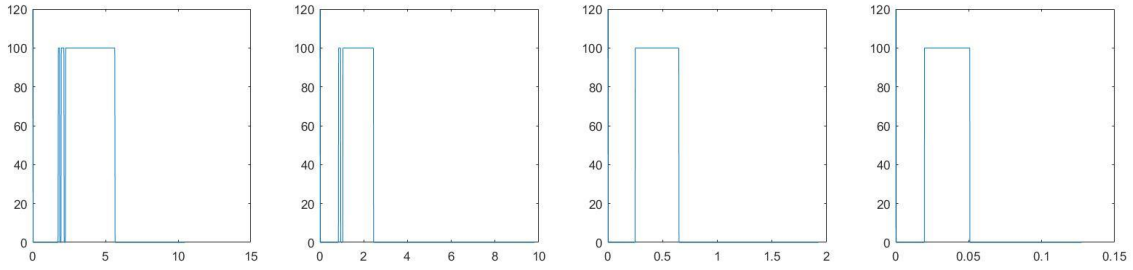
เป็นส่วนของการวาดกราฟฟิค เพื่อแสดงเลน และ**การหาตำแหน่งจุดกึ่งกลางของเลน** โดยมี Output เป็นสัญญาณรูปภาพที่วาดเส้นของเลน และค่าจุดกึ่งกลางของเลน

<u>การทดลองการตรวจจับสัญญาณไฟจราจรเพื่อหยุดการขับเคลื่อนของรถยนต์</u>

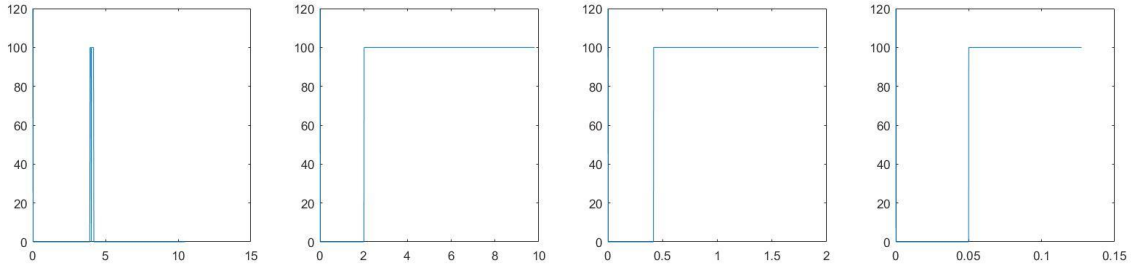
ในการทดลองนั้น เราได้เก็บค่าการเคลื่อนที่ของรถและการตรวจจับไฟจราจรมาโดยการกำหนดให้เมื่อตรวจพบสัญญาณไฟแดง จะทำการเก็บค่าเป็น 100 โดยหากไม่เจอจะเป็น 0 และเมื่อรถยนต์เกิดการเคลื่อนที่ จะเก็บค่าเป็น 100 และหยุดนิ่งเป็น 0 เช่นกัน เนื่องจากการเก็บค่าความเร็วไม่เสถียรและสิ่งที่เราสนใจในการควบคุมรถจากโจทย์ที่กำหนดให้คือต้องหยุดนิ่งก่อนถึงเส้นขาวที่ไฟแดง ดังนั้นจึงเก็บค่าเพียงแค่การเคลื่อนที่ ไม่ลงละเอียดถึงความเร็ว

โดยเราได้ออกแบบการทดลองโดยการเปลี่ยนความถี่การ Sampling ของสัญญาณต่างๆในระบบ ทั้งของภาพ , การส่งข้อมูล และการรับค่าจากพินต่างๆ โดยได้ผลดังนี้

การเคลื่อนที่ของรถ



การตรวจจับไฟจราจร



Sampling rate : 1/50 1/100 1/1000 1/10000 second unit

ผลการทดลอง

จะสังเกตได้ว่าเมื่อมี Sampling rate ต่ำ การตรวจจับภาพช้าจะทำให้การส่งข้อมูลเพื่อหยุดการทำงานของรถช้าลงด้วย ดังนั้น เมื่อเพิ่มความเร็วของการ Sampling จะได้ผลลัพธ์ซึ่งก็คือการตอบสนองของระบบขับเคลื่อนต่อไฟจราจรนั้นเป็นไปได้อย่างรวดเร็วมากยิ่งขึ้น สังเกตจากขอบขาขึ้นของสัญญาณการตรวจจับไฟจราจร และขอบขาลงของการเคลื่อนที่ของรถ

Sampling rate 1/50 second : ได้ผลลัพธ์ว่าการตรวจจับไฟจราจรนั้น ไม่สามารถทำให้รถหยุดก่อนไฟแดงได้ สังเกตได้จากขอบขาลงของกราฟการตรวจจับไฟจราจร ที่มีขอบขาลงก่อนที่รถจะหยุดการเคลื่อนที่ ซึ่งหมายความว่าหากเหยียบคันเร่งต่อไป รถจะสามารถขับเคลื่อนต่อไปได้

Sampling rate 1/100 second : ได้ผลลัพธ์ว่าการตรวจจับไฟจราจรนั้น สามารถทำให้หยุดการทำงานของรถยนต์ได้ทันก่อนที่จะหลุดออกจากไฟแดงไป โดยรถจะหยุดหลังจากที่ตรวจจับไฟจราจรได้ประมาณ 0.5 วินาที

Sampling rate 1/1000 second : ได้ผลลัพธ์ว่าการตรวจจับไฟจราจรนั้น สามารถทำให้หยุดการทำงานของรถยนต์ได้ทันก่อนที่จะหลุดออกจากไฟแดงไป โดยรถจะหยุดหลังจากที่ตรวจจับไฟจราจรได้ประมาณ 0.25 วินาที

Sampling rate 1/10000 second : ได้ผลลัพธ์ว่าการตรวจจับไฟจราจรนั้น สามารถทำให้หยุดการทำงานของรถยนต์ได้ทันก่อนที่จะหลุดออกจากไฟแดงไป โดยจากกราฟจะสังเกตได้ว่าขอบขาขึ้นและขอบขาลงนั้น อยู่ในระยะที่ใกล้เคียงกันจนเกือบจะทับกัน ซึ่งหมายความว่าเมื่อกล้องตรวจจับไฟจราจรได้แล้ว รถก็สามารถหยุดได้ทันทีที่ตรวจจับได้เลย

โดยในการทดลองนั้น จะพบว่าเมื่อใช้ Sampling rate ที่สูงขึ้น จะทำให้ความเร็วของการประมวลผลข้อมูลภายในคอมพิวเตอร์ต่ำลง แต่จะช่วยเพิ่มประสิทธิภาพและการตรวจจับสัญญาณไฟเพื่อนำไปสั่งการรถได้เร็วมากยิ่งขึ้น

<u>สรุปผลการทดลอง</u>

จากผลการทดลองสามารถสรุปได้ว่า เมื่อ Sampling rate มีค่าสูงขึ้น ระบบจะสามารถตรวจจับไฟจราจรได้มีประสิทธิภาพมากยิ่งขึ้น และในทางกลับกัน เมื่อเพิ่ม Sampling rate สูงขึ้น การประมวลผลภาพและการทำงานในส่วนต่างๆของโปรแกรมก็มีประสิทธิภาพที่ต่ำลง ซึ่งจะสังเกตได้จากเวลาในการ Simulation ภายในโปรแกรม ซึ่งจะเดินช้าขึ้นเรื่อยๆเมื่อเทียบกับเวลาปกติ