Model-Based Design for NXP Freescale Cup Car

vDeSi, Department of Electronic and Telecommunication Engineering
King Mongkut's University of Technology Thonburi
February 7, 2560 BE

Abstract

This paper shows how students used the modeling and simulation capabilities of the Matlab/Simulink to establish and implement the control design for the AMAS (Annual Student Meeting Automotive Embedded System) project. Creating and simulating model gives us a better understanding of the processes and almost bug-less transfer of the code to the embedded processor. Model was used also for the first estimation of a device controller.

Keywords Freescale cup, Model-based Design, Automotive Design, Student Competition, Image Processing

1. Introduction

The AMAS project is a global engineering multilevel competition where student teams build, program and race an intelligent autonomous model car around tracks which have points on each track. The competition aims to deepen student's knowledge about embedded control systems design using Model-Based design and Advanced Driver Assistance System via automatic break.

During the design phase students must tackle several Science, Technology, Engineering, and Math (STEM) related issues such as embedded microcontroller programming, communication control, modeling and implementation, as well as overall vehicle dynamics (physics). Soft skills are also trained through team collaboration, communication and project management.

Instead of detailed algorithm description, in this paper we try to show how to use a model based design and intuitive approaches for successful implementation of the controller algorithm.

Brief introduction into this competition is sketched in the Sect. 2. In the Sect. 3 we show the electronic design, creation of the whole car model and the controller. In the last section we summarize our experiences, recommendations and planning.

2. Running Competition Overview

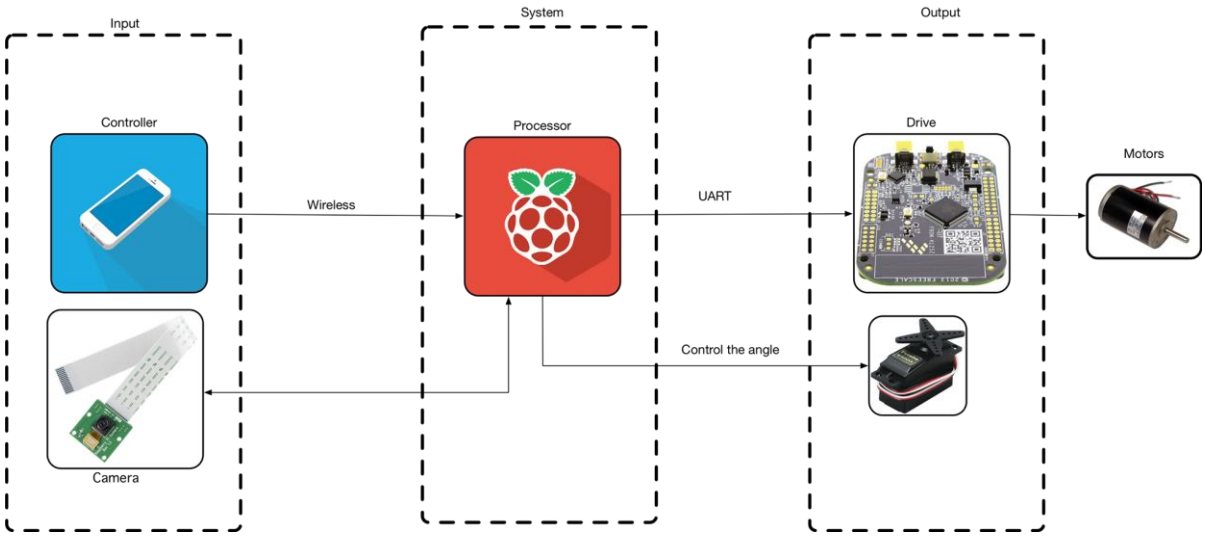
Running Competition is separated to 6 tracks: Speed Ring, Stop, Double Victory Monument, Dead-end Soi, Paragon Pit Stop and Rural Traffic Light. In the speed ring, the fastest car without any accident will get full score. The others track's point is depend on how you can pass the track. Finally, for the Rural Traffic light, the car must execute the automatic break at red traffic light then car must completely stop in front of the white line before traffic.

3. Car Modelling

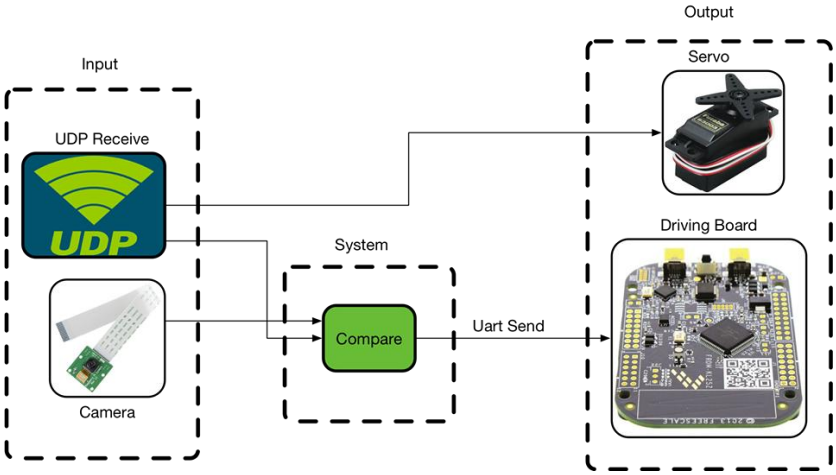
3.1 Schedule Plan

Elements	2016						2017							
	Aug		Sep		Oct		Nov		Dec		Jan		Feb	
1. AMAS-MBD Seminar and Workshop														
2. Consider the topic and analyze problems														
3. Study and gathering resources.														
4. Structure designing														
5. Implementation														
6. Testing and debugging														
7. Development														
8. Report and documentation														
9. Presentation														

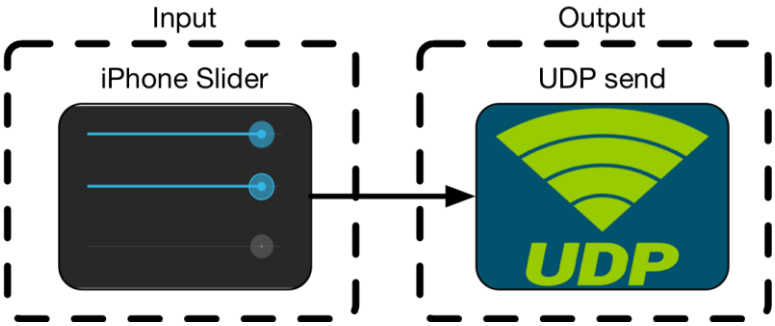
3.2 Overall system architecture



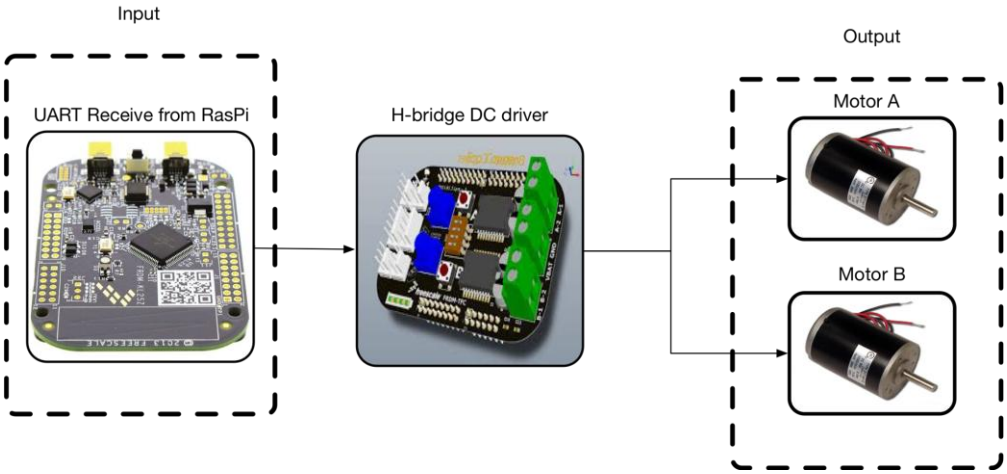
The overall system architecture can be divided into three main parts which is collected the data from 2 inputs by controller (wireless communication) and camera (wired communication).The system will make a process and decision that meets the conditions. After finishing process, the system will send the data to the output to control the movement of the car by connecting the output to driving board and servo using UART communication and GPIO respectively.



The communication path is used by UDP and UART. UDP is sent from the controller to control the angle of servo and UART is sent out from the system to the driving board to control speed of DC motor.



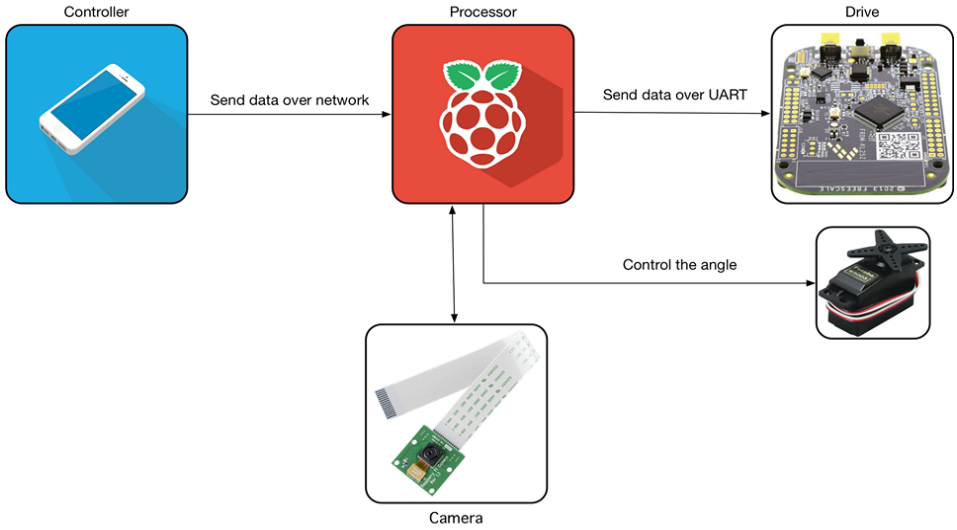
The input from the controller is sent by iPhone Slider and go directly through the output by using UDP as the wireless communication to control servo.



The data is sent from driving board to H-bridge DC driver to control DC motor A and B ,by using wired communication.

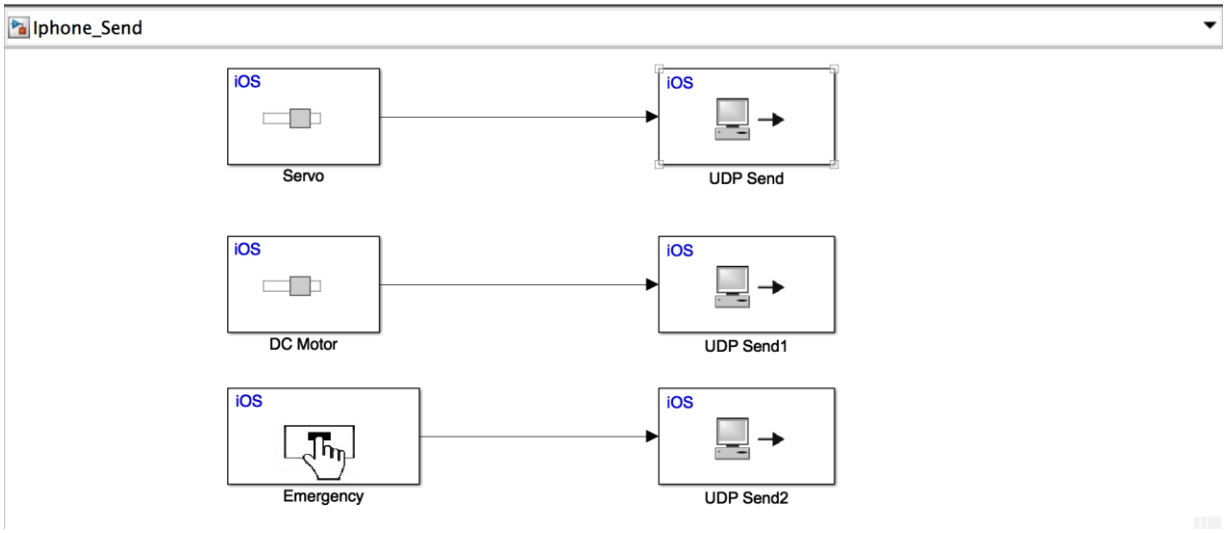
3.3 Overall system modelling

For successful design it was necessary to understand not only the basic physics beyond the car and the properties of the proposed controller, but we also need to know how certain parameter influences another and which of them are the most important factors.



Basically, before we go through deeper in this model, we need to understand the overall system architecture of this model car. There are 3 main cores of this car. Firstly, it is the controller which used to control the car to turn left or turn right, going forward or going backward and break. Secondly, it is the processor part. Its duty is to receive the data from the controller and process the digital image from the camera for the automatic break, then send the data to drive and steering. Finally, the drive component which used to drive the dc motor from the processor signal.

3.3.1 Controller Design model

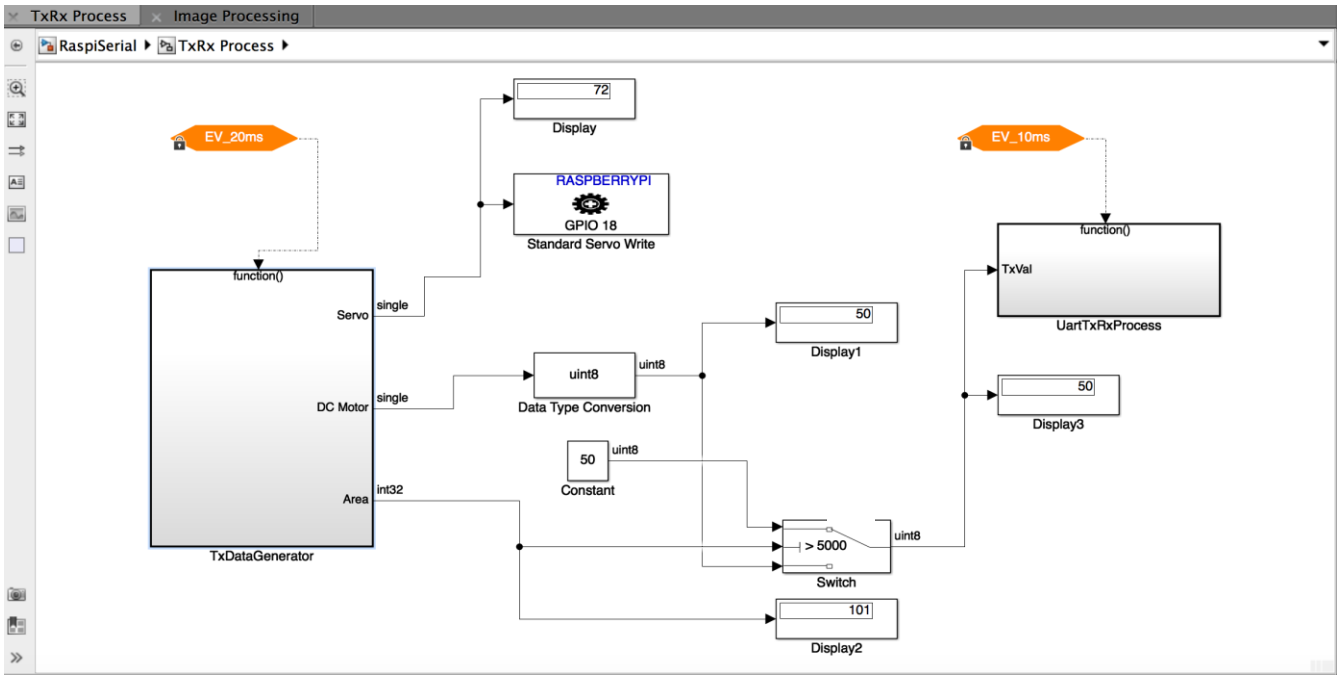


The controller that we use is iPhone because in Matlab/Simulink provide the hardware support package that support in the MacOS and Xcode 7. The support package includes a library of Simulink blocks for accessing the device's sensors, capturing and playing audio and video, creating UI widgets and communicating with other devices through a network interface.

In this model, we send 3 parameters that used to manipulate the car. First is the Servo which send the angle of the car. Its range is about 40 to 100 degree and the center is 60 degree. Second is the DC motor which send the signal that control the speed of the car. Its range is about 30 to 70 and the center is 50. We decide to use the slider function to handle these 2 parameters. Finally is the button that handle break function of the car. The button send the Boolean logical which is 1 when it is pressed and 0 when there is no operation.

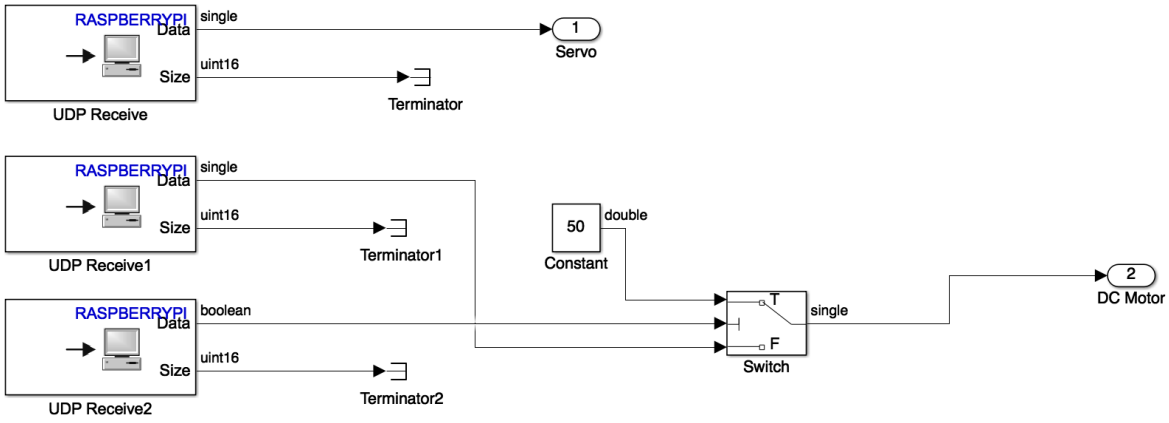
For these parameters, we send it over the User Datagram Protocol (UDP) which is one of the core members of the Internet protocol that suite to the Raspberry Pi. This block need to set the Remote IP address and Remote IP port parameters to the IP address and port number of the receiving UDP host, respectively.

3.3.2 Processor Design model



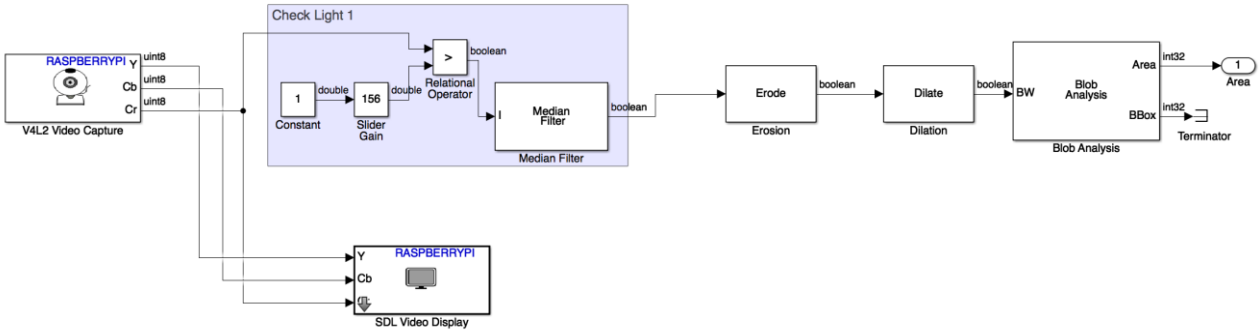
In the processor model, we choose to implement in the Raspberry Pi which is the microcontroller running in Raspbian Pixel operating system. This processor model is include the receiving data which is generated from iPhone and the ADAS which is the automatic break. First we need to look at the TxDataGenerator block because this block has a received function and the automatic break which provide the signal of Servo, DC motor and Area.

In the TxDataGenerator block include the UDP receive and the image processing system.



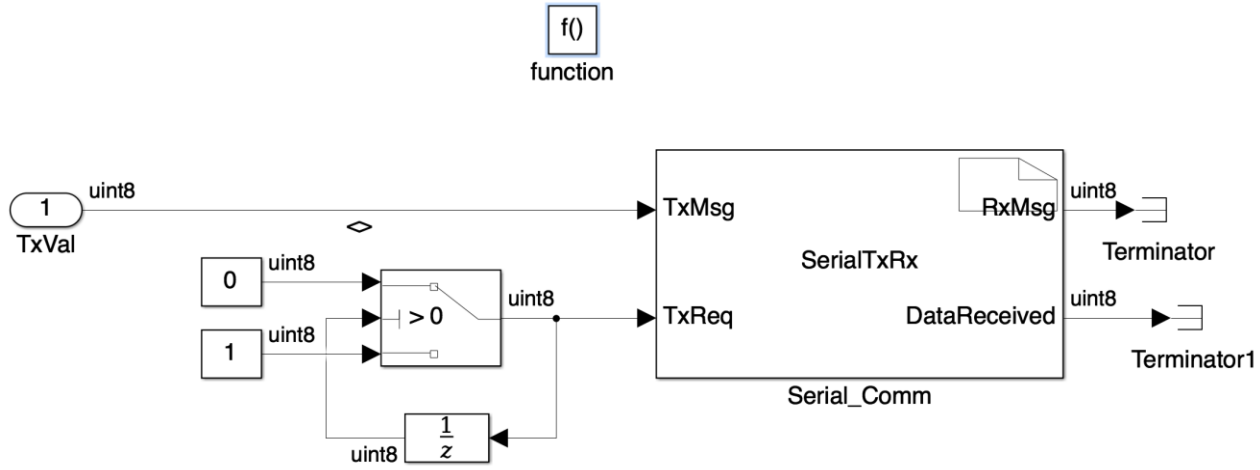
The received function which is provided by the Raspberry Pi hardware support packages receive 3 UDP packages from another UDP host in an Internet network. UDP Received block collect servo data, UDP Received1 block collect DC motor data and UDP Received2 block collect data from the emergency button .The sending UDP host have to send UDP packages to the Local IP port specified in the block properties. The Switch block compares the data from the iOS button which pass through the 50 when the button is pressed, otherwise it send the data from UDP Received1.

After this, we send the Servo's signal to the Standard Servo Write block which is provided from Raspberry Pi hardware support package to control the angle of the car.

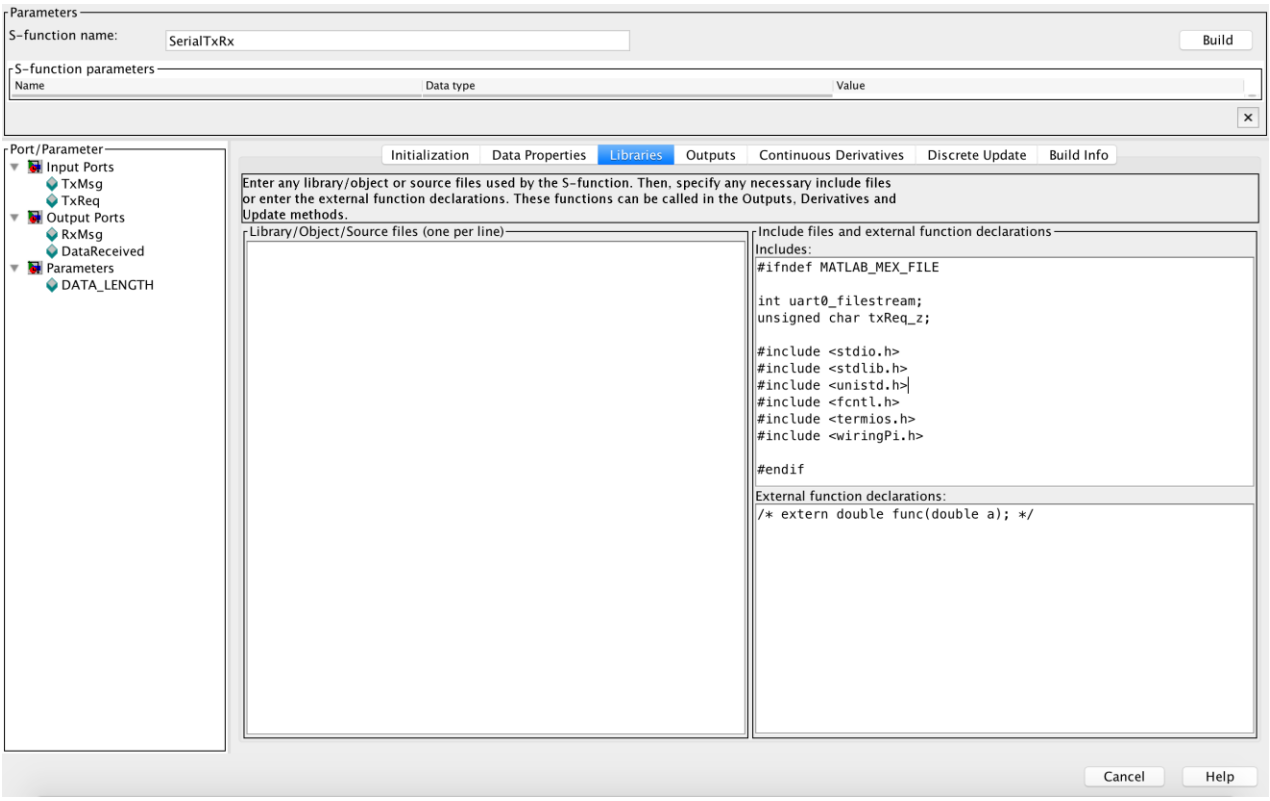


The image processing section is used for automatic break. This block use the image processing toolbox and Raspberry Pi hardware support package. The beginning of algorithm, we have to receive the image from the Raspberry Pi through the V4L2 Video Capture block. Then we choose only Cr component to process the red component in this image by converting to the binary image and put it into the Median Filter. After this, we use erosion and dilation to enhancement the image. Last but not least, we use Blob Analysis to tell the maximum area of red component in the image.

After analyzing the TxDataGenerator block, we will go deeper through this model about sending the data. First, we send the servo data through the GPIO which is the GPIO18 via the Standard Servo Write block. Secondly, we send the area and DC motor data to the Switch block which pass through the 50 when the area is in the condition. Therefore it passes from the dc motor to the UartTxRxProcess block when the emergency button is not pressed.



This is inside the UartTxRxProcess block written by engineer from Toyota Tsusho Electronics Thailand. This block use the WiringPi library which is the GPIO interface library for the Raspberry Pi. In the Serial_Comm block is the s-function builder which receive the TxMsg which is the transited message and TxReq, then send the RxMsg and received data. Now, let's go deeper in this block.



This is library file. First we have to define the MATLAB_MEX_FILE that is used to build the C code in the MATLAB and declare 2 variables: uart0_filestream which is the integer and txReq_z which is the character. Then, we include the library that we use to model this car.

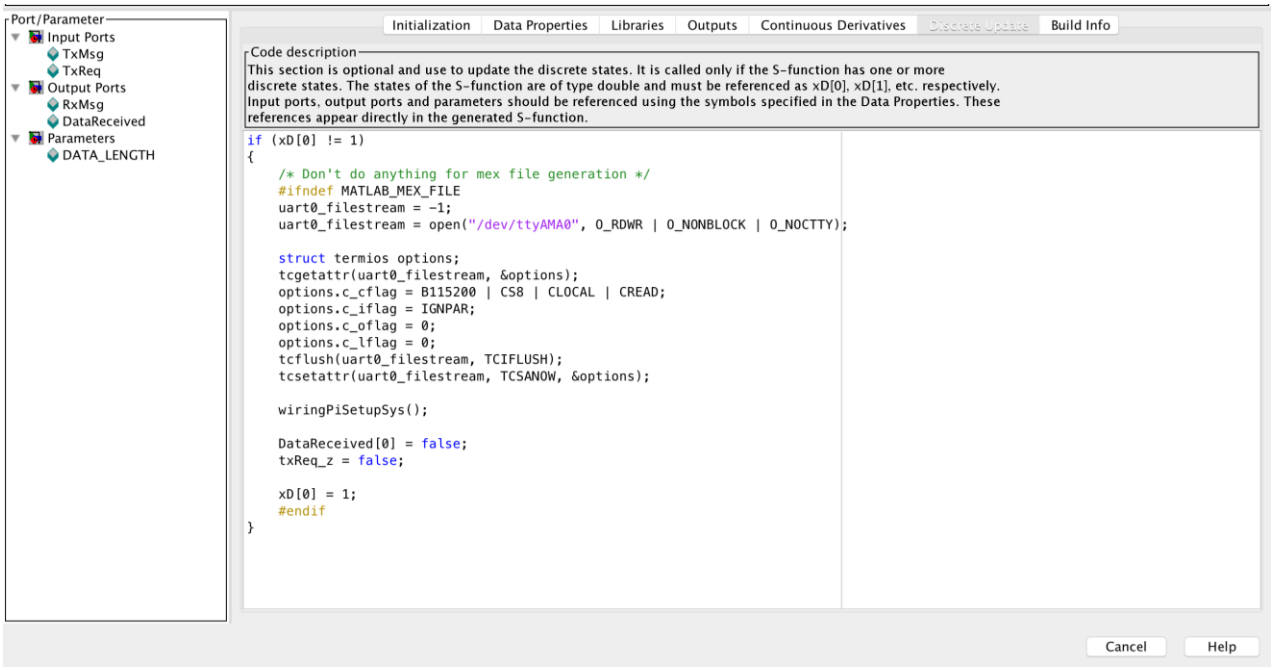
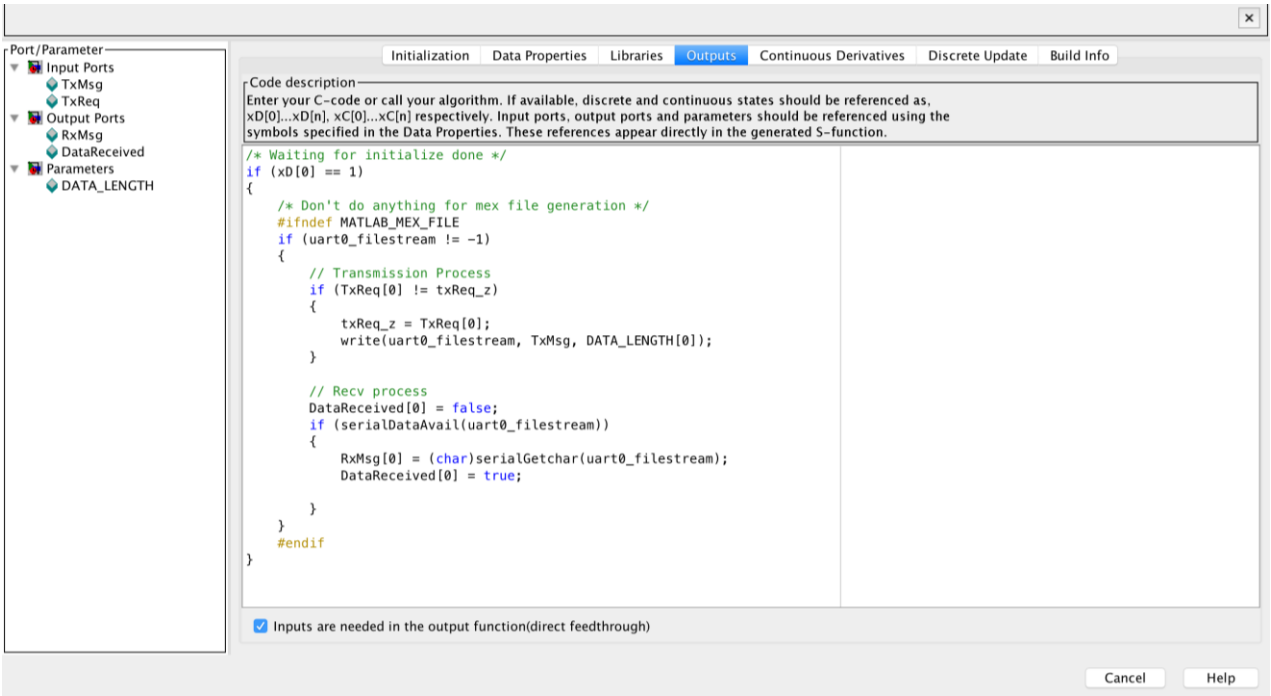
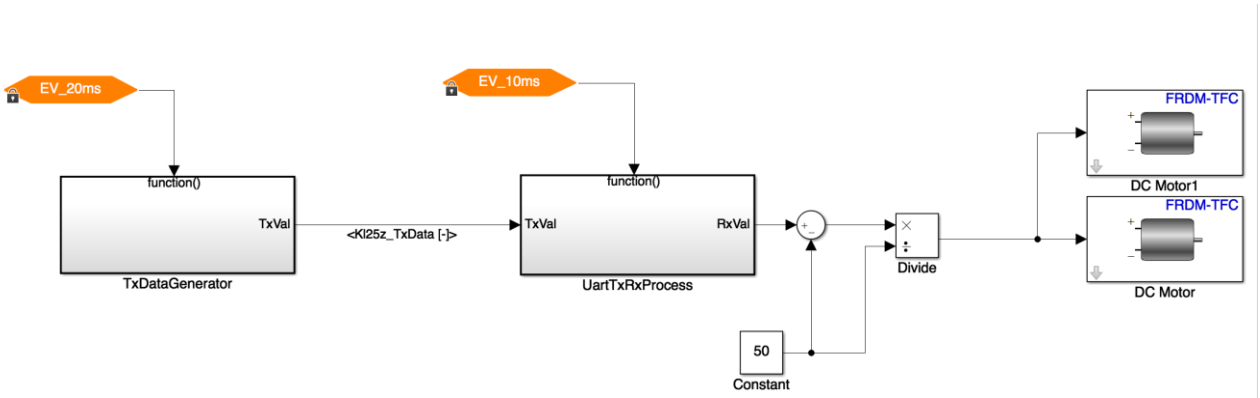


Fig. This is the custom code section in the s-function builder.

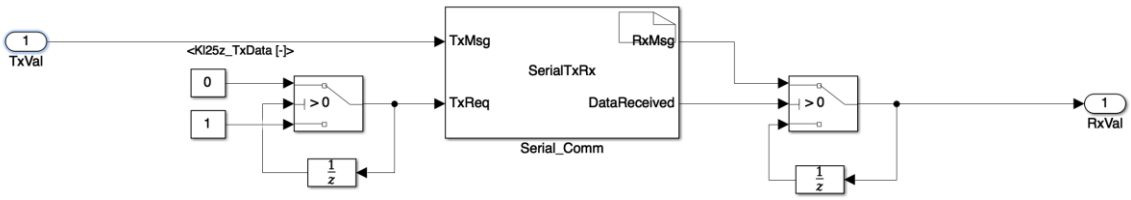


In conclusion of the process section, we send the Servo data to the Standard Servo Write block and sense the red traffic light by using the image processing to compare with the area threshold and send the DC motor data to the drive section through UART serial communication.

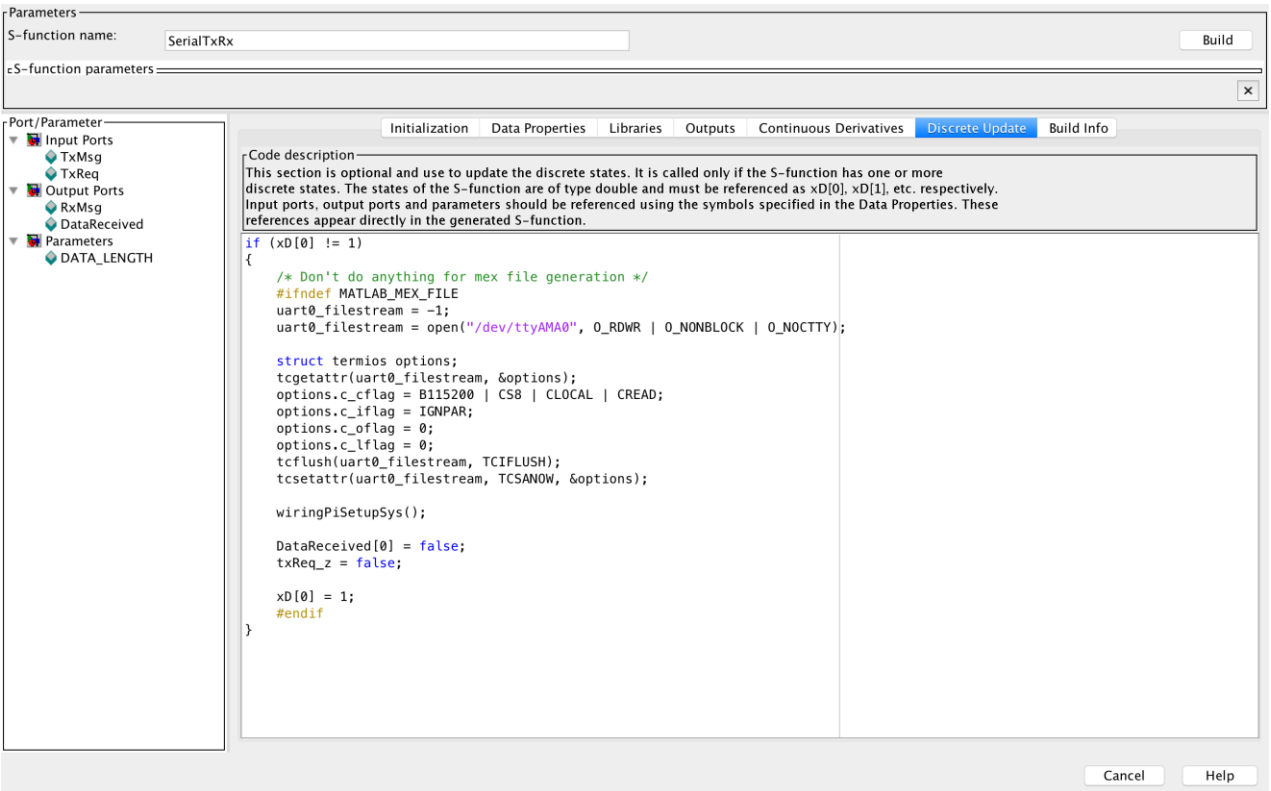


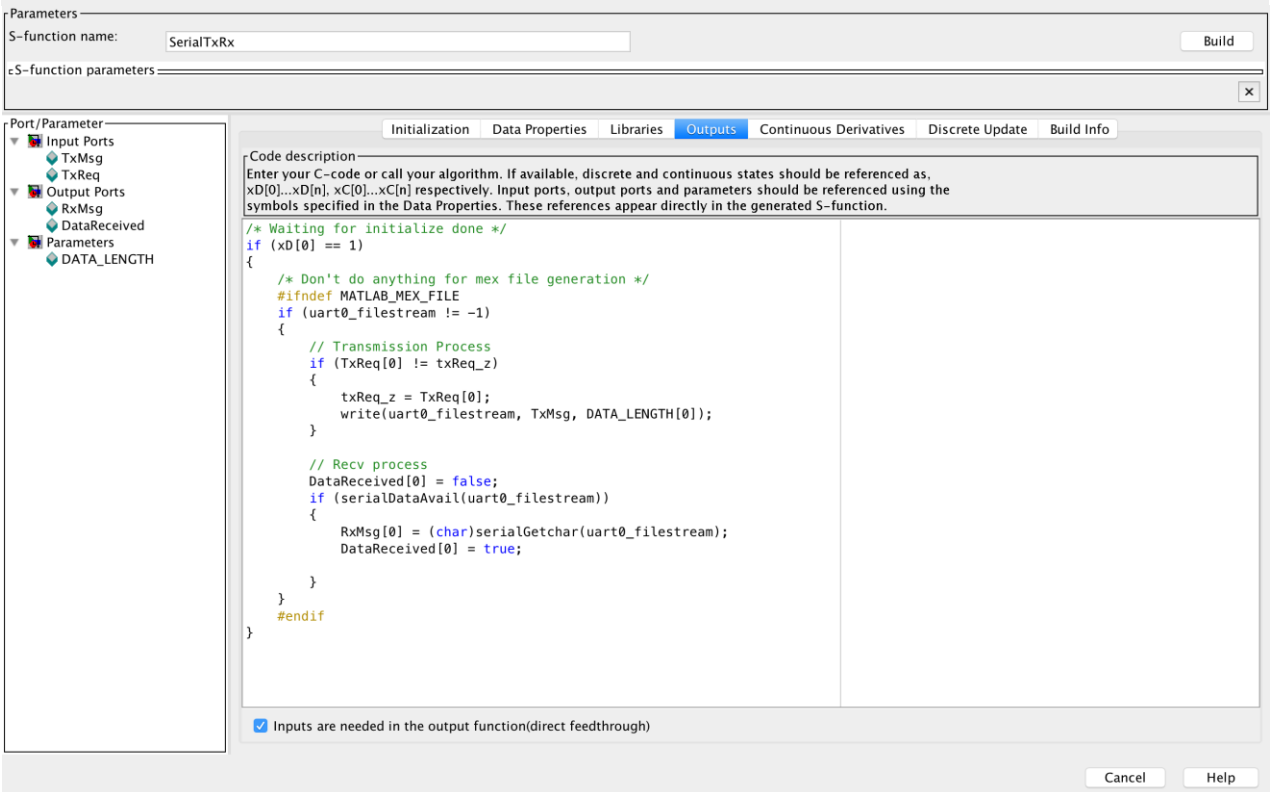
3.3.4 Driving Design model

In the driving section, we implement on the FRDM-KL25Z and the shield FRDM-TFC which is established by the NXP semiconductor. For the model, there are the TxDataGenerator which is used to send the data and the UartTxRxProcess which is used to receive the data that generated from Raspberry Pi. These blocks is written by engineer from Toyota Tsusho Electronic Thailand. In this case, we do not use the TxDataGenerator. Therefore, it only receive the DC motor that is range 0-100 and send it to the DC Motor block which is provided by FRDM-KL25Z Hardware Support Package which its input accepts the normalized values (-1 to 1, full reverse to full forward). The constant 50 is the number that is calculated for the normalized value by subtract with the input data and then divided by 50 to get -1 to +1.



This is what inside the UartTxRxProcess block which is the serial communication using s-function builder.


Parameters
S-function name:
SerialTxRx
Build
S-function parameters
Port/Parameter
Input Ports
TxMsg
TxReq
Output Ports
RxMsg
DataReceived
Parameters
DATA_LENGTH
Initialization
Data Properties
Libraries
Outputs
Continuous Derivatives
Discrete Update
Build Info
Code description
This section is optional and use to update the discrete states. It is called only if the S-function has one or more
discrete states. The states of the S-function are of type double and must be referenced as xD[0], xD[1], etc. respectively.
Input ports, output ports and parameters should be referenced using the symbols specified in the Data Properties. These
references appear directly in the generated S-function.
if (xD[0] != 1)
{
/* Don't do anything for mex file generation */
#ifndef MATLAB_MEX_FILE
uart0_filestream = -1;
uart0_filestream = open("/dev/ttyAMA0", O_RDWR | O_NONBLOCK | O_NOCTTY);
struct termios options;
tcgetattr(uart0_filestream, &options);
options.c_cflag = B115200 | CS8 | CLOCAL | CREAD;
options.c_iflag = IGNPAR;
options.c_oflag = 0;
options.c_lflag = 0;
tcflush(uart0_filestream, TCIFLUSH);
tcsetattr(uart0_filestream, TCSANOW, &options);
wiringPiSetupSys();
DataReceived[0] = false;
txReq_z = false;
xD[0] = 1;
#endif
}
Cancel
Help


Parameters
S-function name:
SerialTxRx
Build
S-function parameters
Port/Parameter
Input Ports
TxMsg
TxReq
Output Ports
RxMsg
DataReceived
Parameters
DATA_LENGTH
Initialization
Data Properties
Libraries
Outputs
Continuous Derivatives
Discrete Update
Build Info
Code description
Enter your C-code or call your algorithm. If available, discrete and continuous states should be referenced as,
xD[0]...xD[n], xC[0]...xC[n] respectively. Input ports, output ports and parameters should be referenced using the
symbols specified in the Data Properties. These references appear directly in the generated S-function.
/* Waiting for initialize done */
if (xD[0] == 1)
{
/* Don't do anything for mex file generation */
#ifndef MATLAB_MEX_FILE
if (uart0_filestream != -1)
{
// Transmission Process
if (TxReq[0] != txReq_z)
{
txReq_z = TxReq[0];
write(uart0_filestream, TxMsg, DATA_LENGTH[0]);
}
// Recv process
DataReceived[0] = false;
if (serialDataAvail(uart0_filestream))
{
RxMsg[0] = (char)serialGetchar(uart0_filestream);
DataReceived[0] = true;
}
}
#endif
}
Inputs are needed in the output function(direct feedthrough)
Cancel
Help

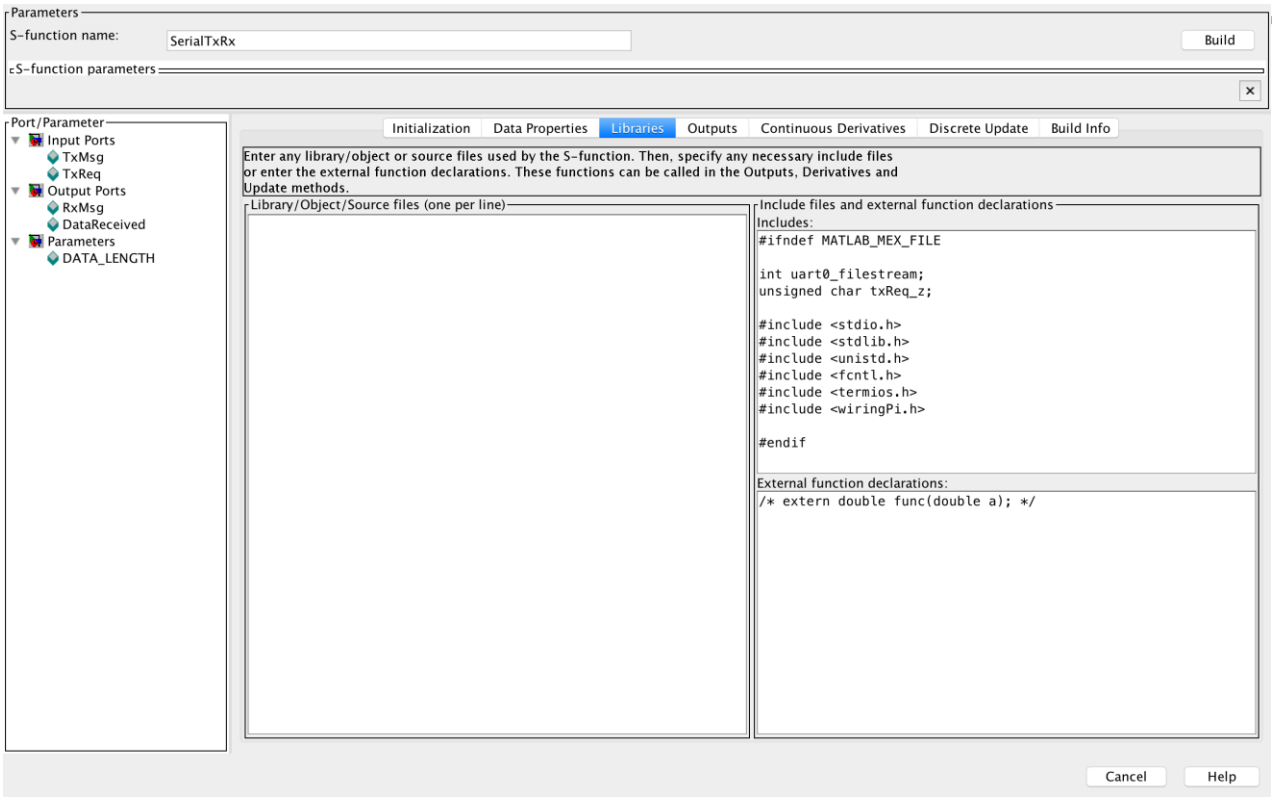


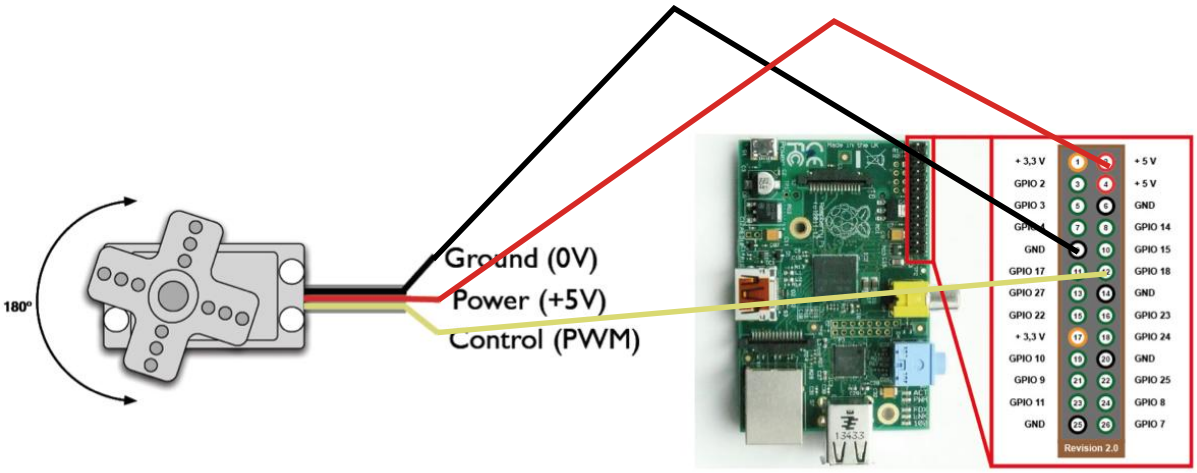
Fig. This is the custom code that we put it into the s-function builder.

In summary, the duty of the FRDM-KL25Z and FRDM-TFC is only receive the data from the processor (Raspberry Pi) and send the data to drive the motor which is connected to the FRDM-TFC.

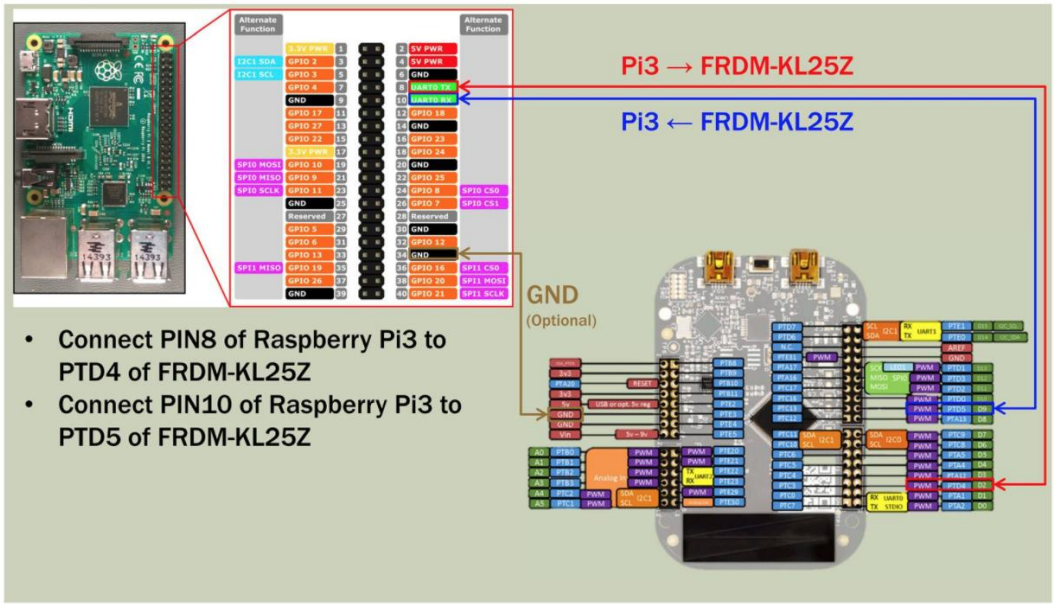
3.4 Overall Electronic Design

For all electronic design, we are concerning about the Raspberry Pi, FRDM-KL25Z and FRDM-TFC.

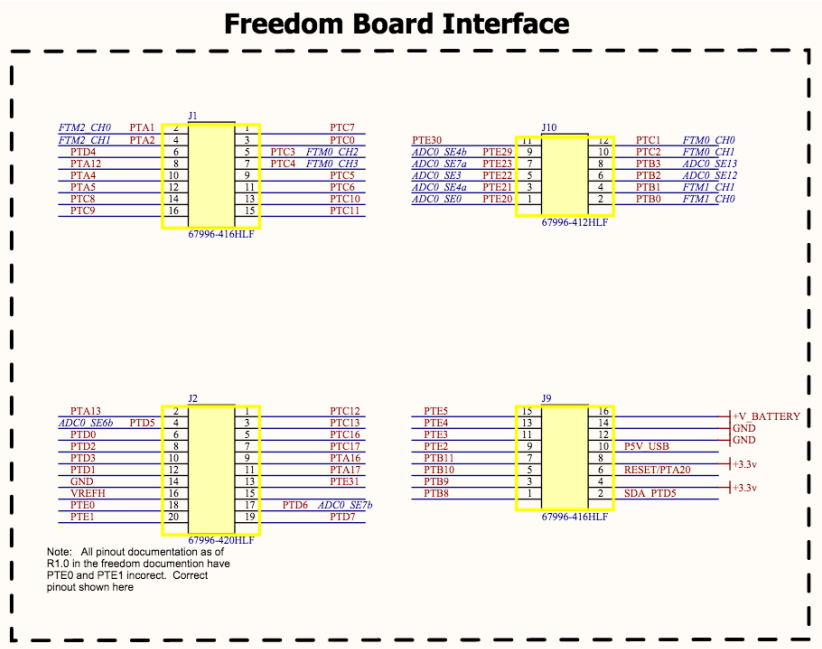
- In this model, Raspberry Pi is connected with the Servo motor and The PWM is connected to GPIO18.



- For the serial communication model between the Raspberry Pi and the FRDM-KL25Z, we connect GPIO14 of Raspberry Pi to PTD4 of the FRDM-FL25Z and connect the GPIO15 of the Raspberry Pi to PTD5 of FRDM-Kl25Z



- After that, we are trying to connect dc motor port between FRDM-KL25Z and FRDM-TFC by looking into the schematic diagram below.



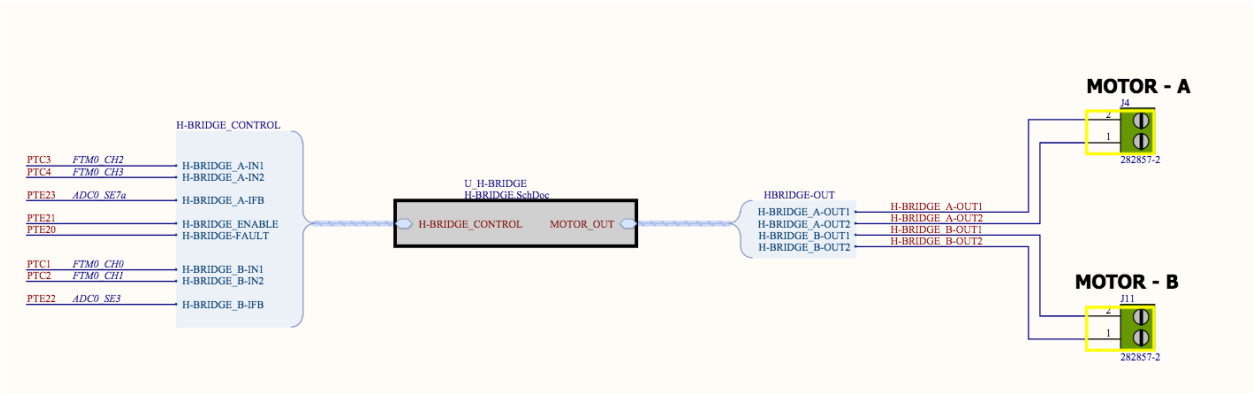
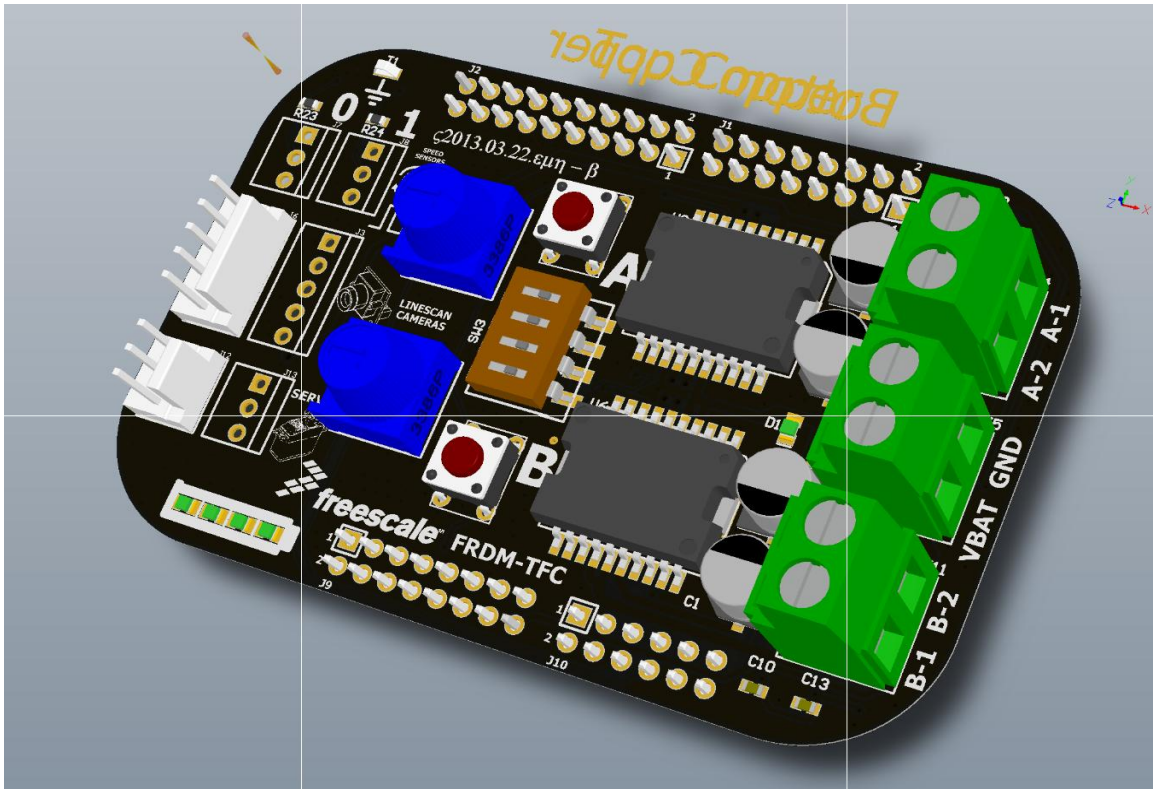


Fig. The schematic diagram of the FRDM-TFC which is connected to the H-Bridge control.

Therefore, the port that deal with the motor are PTC3, PTC4, PTE23, PTE21, PTE20, PTC1, PTC2, and PTE22. Then look into the Freedom Board interface, so we connect those pins directly from FRDM-TFC to FRDM-KL25Z.

The picture below illustrate where are the pin-out of the FRDM-TFC.



After, we connect with the port of the FRDM-TFC, then we directly connect with the FRDM-KL25Z by wire. Finally, we connect the battery with the VBAT and GND, then connect 2 motors with the A port and B port respectively.

3.5 Testing Software

In this model, you will see the display block that we create for testing the value of the data and delay of the system. The result is satisfied with the serial communication in transited and received process but for the UDP sending, there was a little bit delay in some period of time because of the iPhone's application. So, we solve this problem by restart the application in the iPhone.

Before we build the model, we check for the signal from the oscilloscope. It means that the Servo motor's pin GPIO18 is sent the pulse-width modulation signal and the dc motor is sent the amplitude depend on its speed.

4 Conclusion

Key to the success in our case was the ability and passion to spend many hours with testing and redesigning the car again and again. The task seems to be an easy, but for students, especially in their third year at the university this is not true. They have to fight with a new software packages, their installation, mastering the development cycle, and with all the hardware issues at the same time. Especially when the team choose to design also their proprietary hardware, the time constraints were crucial. Otherwise, probably the main challenge was to implement the image processing routines into the embedded microcontroller.

Experiences: Creating and simulating the model gives us (a) better understanding of the processes and (b) almost bug-less transfer of the code to the embedded processor and (c) first estimation of the controller parameters.

Model is not perfect, but even in this form it is better than nothing. It helped us to understand the behavior of the car, which physical quantities are important more and which are less. This leads to understanding, what can be neglected and what is important. It is a big step over the trial-and-error approach often applied by students.

Racing is a challenge that virtually every human knows, has no language barriers, and never fails to provide excitement, adrenaline and with it a platform to educate.

Acknowledgements This project has been supported by Toyota Tsusho Electronics Thailand, vDesi Laboratory and Department of Electronic and Telecommunication Engineer, KMUTT.

References

1. The Freescale: The Freescale Cup 2014 EMEA Rules v 1.2 11 (2014). https://community. freescale.com/docs/DOC-94949
2. McLellan,J.,Mastronardi,A.:Engaging students: the growing smart car competition.In:2009 Annual Conference and Exposition, Austin, Texas (2009). https://peer.asee.org/4983
3. Zhicong, S., Xuemei, L., Mei, C., Hongbin, Z.: The design of smart car based on Freescale processor. In: 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering (CCTAE), Jun 12, vol. 2, pp. 508–510. IEEE (2010)
4. Wang,Z.,Liu,Y.:Design of road tracing navigation control for smart car use CCD sensor.In: 2010 International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT), April 17 vol. 1, pp. 345–348. IEEE (2010)
5. Xiuquan, W., Xiaoliu, S., Xiaoming, C., Ying, C.: Route identification and direction control of smart car based on CMOS image sensor. In: ISECS International Colloquium on Computing, Communication, Control, and Management, 2008. CCCM'08, vol. 2, pp. 176–179. IEEE (2008)
6. Mc Lellan, J.: History of the Freescale Cup (2016). https://community.freescale.com/docs/ DOC- 1011
7. Krishnan,R.: Electric Motor Drives: Modeling, Analysis, and Control.PrenticeHall(2001)
8. de Sean, W.: MATLAB and Simulink for Embedded Systems and Robotics. The MathWorks, Inc. (2014). https://www.buffalo.edu/content/www/ccr/support/software-resources/compilers-programming-languages/matlab/_jcr_content/par/download_1/file.res/ MathWorks_MATLAB_and_Simulink_for_LEGO.pdf
9. Richard B., Marek L., Model-Based Design of a Competition Car, Advances in Intelligent Systems and Computing 457. 219-229 (2016).
10. Semiconductor, Freescale. FRDM-KL25Z: Freescale Freedom Development Platform for Kinetis KL14. 15/24/25 MCUs. Documentation. http://www.nxp.com/products/software-and-tools/hardware-development-

tools/freedom-development-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL25Z
11. Kelemenová,T.,Kelemen,M.,Miková,L.,Maxim,V.,Prada,E.,Lipták,T.,Menda,F.:Model based design and HIL simulations. Am. J. Mech. Eng. **1**(7), 276–281 (2013). http://pubs. sciepub.com/ajme/1/7/25/
12. Zhao, Z.: The reform and practice of education mode in multi-disciplinary joint graduation design. Procedia Eng. **15**, 4168–4172 (2011). ISSN 1877-7058. http://dx.doi.org/10.1016/j. proeng.2011.08.782 (2011)
13. László, M.: Navigation and robot control with speed optimalization. Master thesis. Slovak University of Technology in Bratislava (2015). http://www.crzp.sk/crzpopacxe/openURL?crzpID=42928&crzpSigla=stubratislava