ASSUMPTION UNIVERSITY

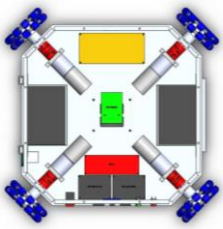
Vincent Mary School of Engineering

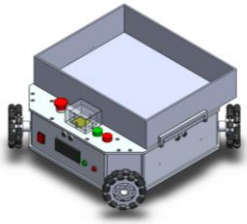
EE 3703: Microprocessor

Project Report

Developing IoT-Controlled robot movement with ESP8266 and Blynk application

"Directional Control for Mobile Robot and LCD Display"





Submitted to : Mr. Sunphong Thanok

Submitted by :
Subgroup 3 (Section 641 and 642)
Group Members :

No.	ID	Name
1.	5715453	Palita Prukhattapong
2.	5728018	Pan Wit Yee
3.	5735234	Thet Phoo Zin
4.	5916962	Pongsa Pongsawakul
5.	5918154	Nyi Nyi Myo Zin Htet
6.	5938101	Min Khant Soe

Semester 1/2019
Date of Submission : 06.12.2019

I. **Project Name** : Directional Control for Mobile Robot and LCD Display

II. Project Objectives

- **Main objective of the whole project:**

- To develop IoT-Controlled robot movement with ESP8266 and Blynk application

- **Main objective for Subgroup 3:**

- To control the direction of the mobile robot
- To show the LCD display

III. Project Description

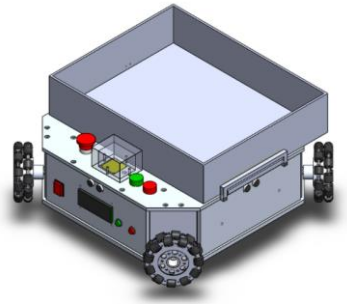
We are taking microprocessor (EE: 3703) class. Our lecturer gave us a class project which is to develop IoT-Controlled robot movement with ESP8266 and Blynk application. The project is divided into 3 Topics and we can choose which part we want to do. Topic 1 is "**RFID Module and Blynk Display**", Topic 2 is "**Communication between ESP8266 and STM32F417IV**", and Topic 3 is "**Directional Control for Mobile Robot and LCD Display**". We decided to choose Topic 3 because we think it will be fun and will help us to get more understanding microprocessor system.

Our task is to make this robot move forward, backward, right and left, and rotate CW and CCW. And we have to show the robot status on LCD screen of robot. We creates the stages (for different directions) and set the time for each stages; it includes 7 stages (Initial, Forward, Backward, Right, Left, CW, CCW).

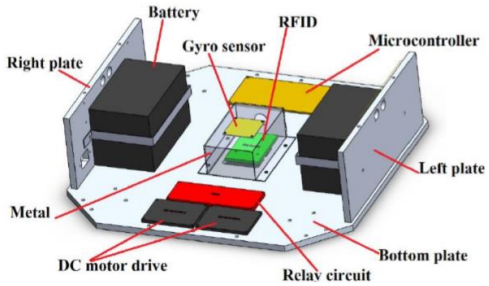
If we press start button, robot will start moving and the direction will be changed stage by stage in sequent. When the final stage (CCW) is completed, it will repeat from (Forward) stage again.

If we press stop button, robot will stop moving and when we press start button again, robot will continue moving at the same stage that it stops. If we press both start and stop buttons together, robot will reset to initial state. The status of robot will be shown on LCD screen.

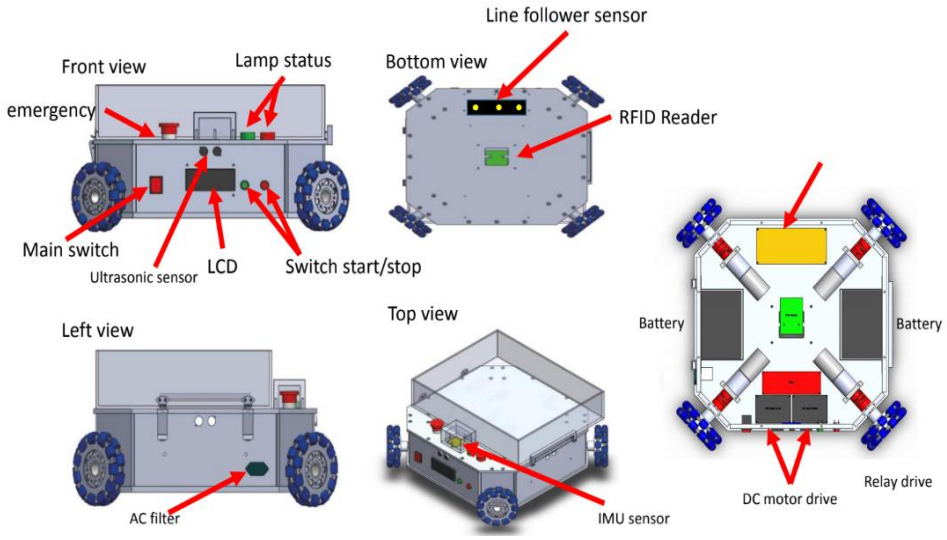
1. Hardware Interfacing



This is the mobile robot that we used for our microprocessor class project.



The hardware devices that are mentioned in above diagram are used inside this mobile robot. Among them we only have to work with dc motor drive, motor, relay, and microcontroller (STM32F417IG).

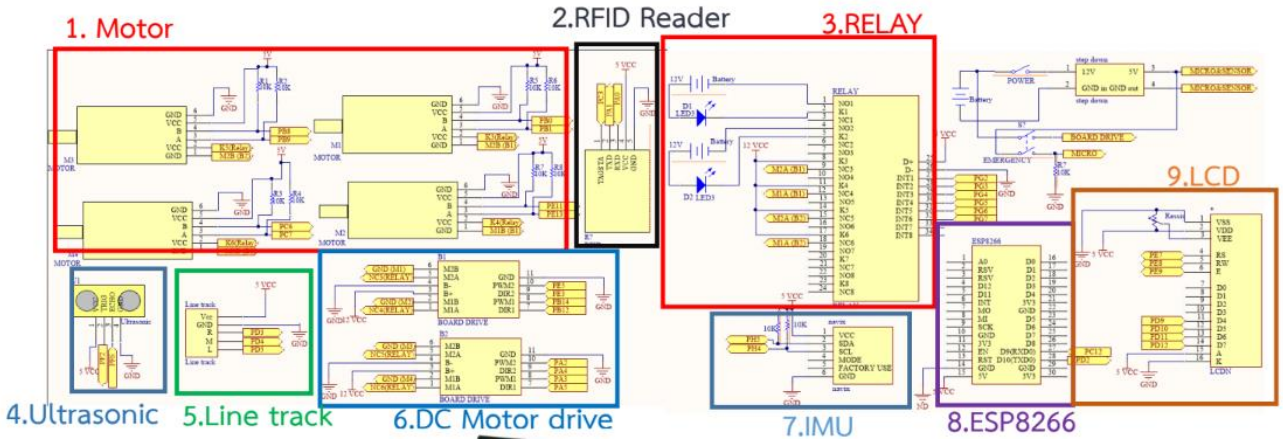


The square shaped (red) switch is the power switch of this robot. LCD is used to display the robot status. The green switch near LCD display is used as the start switch. The red switch beside the green switch is used as the stop switch.

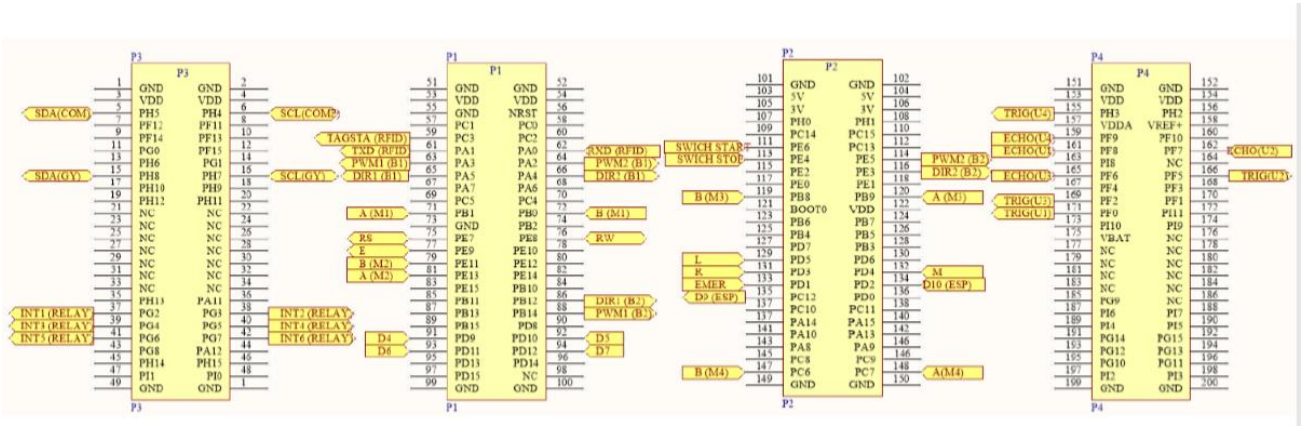
When the robot is stop, the red lamp which is above the robot will be ON. When the robot is moving or operating, the green lamp which is beside the red lamp will be ON.

RFID reader is used to read the robot position. In this project, we don't need to use Line follower sensor.

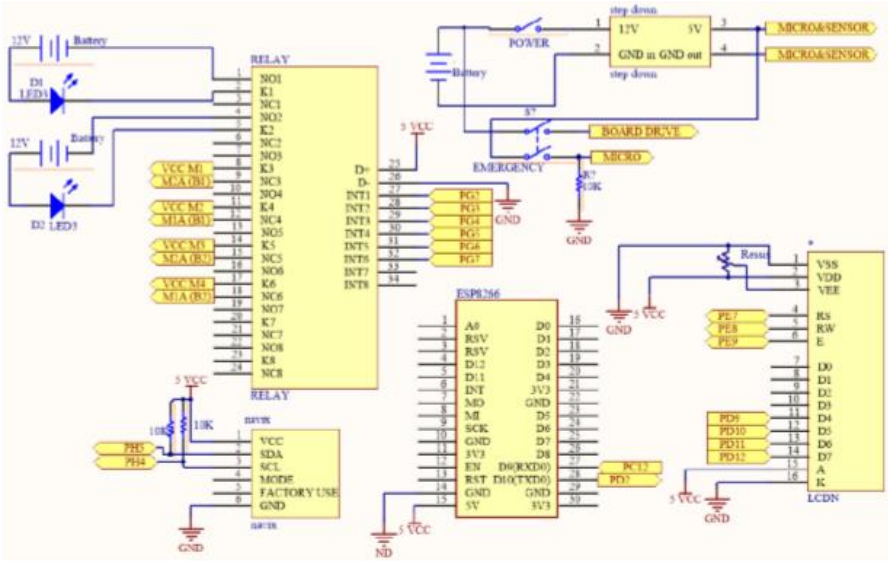
Schematic of the robot



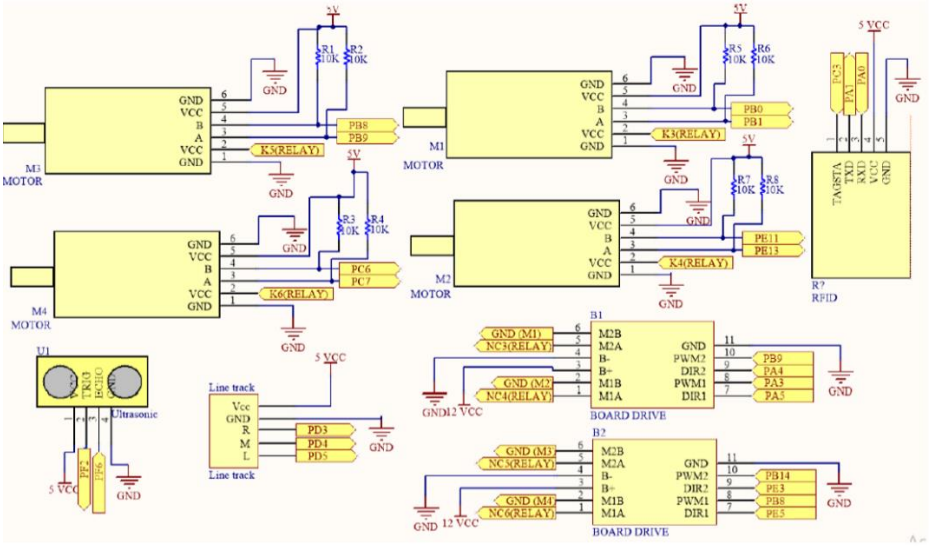
Schematic of STM32F417IG Board



Schematic of LCD, Battery and Relay



Schematic of the motors and drivers



Pin Declaration

Start switch is connected to pin (E4).
Stop switch is connected to pin (E6).
Emergency switch is connected to pin (D1).

Power input pin of the robot is connected to Pin (G2).
Drive 2 pin of the robot is connected to Pin (G3).
Drive 1 pin of the robot is connected to Pin (G4).
Green Lamp pin of the robot is connected to Pin (G5).
Red Lamp pin of the robot is connected to Pin (G6).

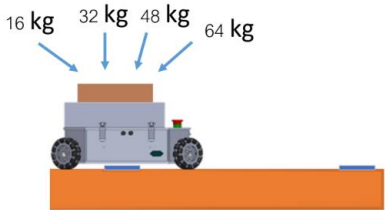
Driver 1 is B1 and Driver 2 is B2 in schematic diagram.

Direction 1 (of Driver-1) pin is connected to pin (A5).
Direction 2 (of Driver-1) pin is connected to pin (A4).
Direction 1 (of Driver-2) pin is connected to pin (E5).
Direction 2 (of Driver-2) pin is connected to pin (E3).

Motor A (PWM 1 of Driver – 1) pin is connected to pin (A3).
Motor B (PWM 2 of Driver – 1) pin is connected to pin (B9).
Motor C (PWM 1 of Driver – 2) pin is connected to pin (B8).
Motor D (PWM 2 of Driver – 2) pin is connected to pin (B14).

Robot Features

- 1.Robot size 450x450x133 mm
- 2.Moving Speed 30 cm/s
- 3.Pay load 50 kg
- 4.Buit-in with RFID TAG



DC motor drive

The DC motor drives are used mainly for good speed regulation, frequent starting, braking and reversing. We know that, normally the rotor of a DC motor is energized by the commutation process through brushes. So, the maximum allowable starting current is determined by the current which can be safely carried out by the brushes without sparking.

DC motor

A DC motor is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

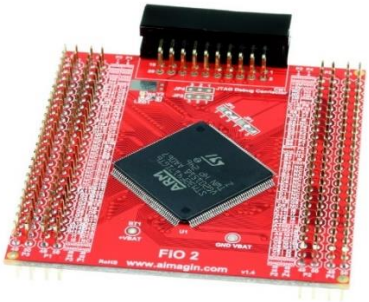
Relay

A relay is an electromagnetic switch that is used to turn on and turn off a circuit by a low power signal, or where several circuits must be controlled by one signal. We know that most of the high-end industrial application devices have relays for their effective working. Relays are simple switches which are operated both electrically and mechanically. Relays consist of an electromagnet and a set of contacts. The switching mechanism is carried out with the help of the electromagnet. There are also other operating principles for its working. But they differ according to their applications. Most of the devices have the application of relays.

LCD Display

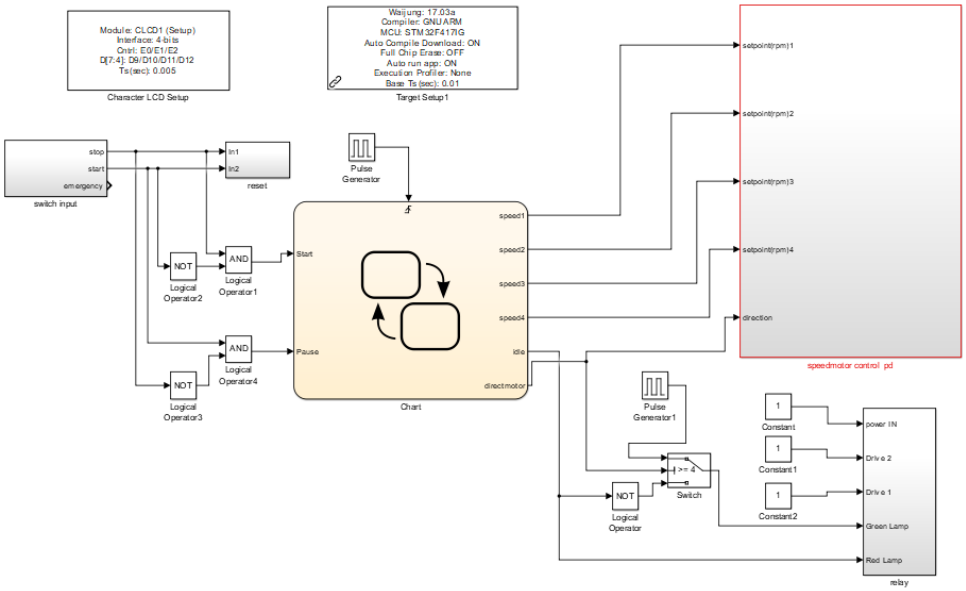
A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made from a matrix of small pixels, while other displays have larger elements. LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement.

STM32F417IG

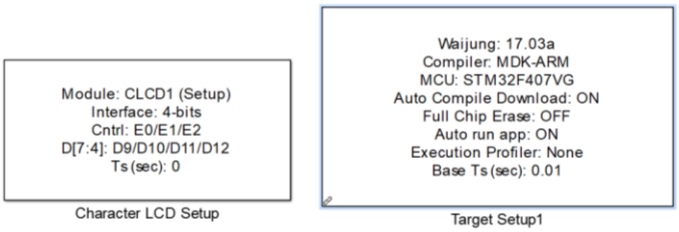


The STM32F415xx and STM32F417xx family is based on the high-performance ARM® Cortex™-M4 32-bit RISC core operating at a frequency of up to 168 MHz. The Cortex-M4 core features a Floating-point unit (FPU) single precision which supports all ARM single precision data-processing instructions and data types. It also implements a full set of DSP instructions and a memory protection unit (MPU) which enhances application security. The Cortex-M4 core with FPU will be referred to as Cortex-M4F throughout this document. The STM32F415xx and STM32F417xx family incorporates high-speed embedded memories (Flash memory up to 1 Mbyte, up to 192 Kbytes of SRAM), up to 4 Kbytes of backup SRAM, and an extensive range of enhanced I/Os and peripherals connected to two APB buses, three AHB buses and a 32-bit multi-AHB bus matrix. All devices offer three 12-bit ADCs, two DACs, a low-power RTC, twelve general-purpose 16-bit timers including two PWM timers for motor control, two general-purpose 32-bit timers. a true random number generator (RNG), and a cryptographic acceleration cell. They also feature standard and advanced communication interfaces.

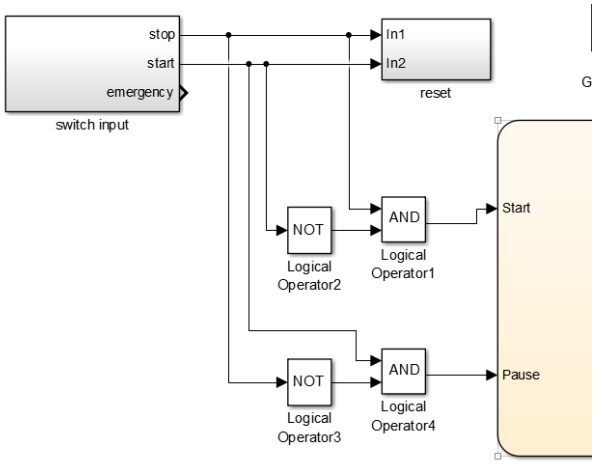
2. Codes and Explanation about codes



This is the overview block set for Directional Control for Mobile Robot and LCD Display of robot.



Character LCD Setup block is used to setup and run the LCD, and Target Setup1 block is used to setup and run the microcontroller board.

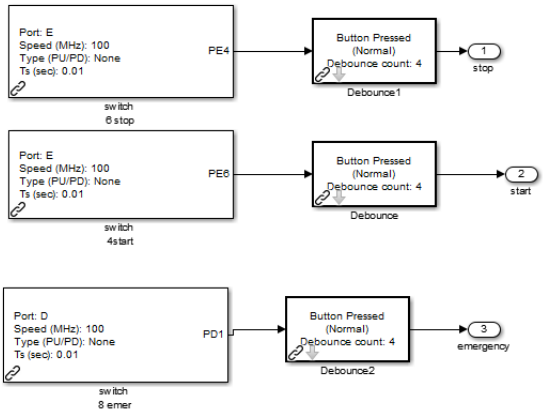


Switch input block is used to receive the signal from Start and Stop switches. Since the switches are ACTIVE LOW switch types if we press them, the input signal will be zero. Therefore, we used **NOT logical Operator 2 and 3** blocks to inverse the start and stop input-

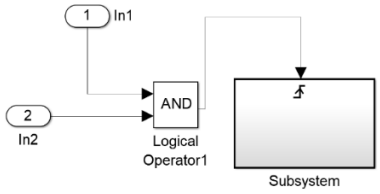
signal respectively. **AND Logical Operator 1 and 2** blocks are used to give the signal to input (Start and Pause respectively) of the Chart.

When start and stop signals are 0 and 1 respectively, the input signal of the **Start** will be 1. Otherwise, it is 0.
When start and stop signals are 1 and 0 respectively, the input signal of the **Pause** will be 1. Otherwise, it is 0.

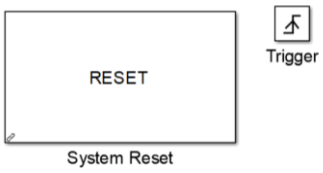
Reset block is used to reset the robot.



This is inside view of the **switch input block**.



This is inside view of the r**eset block**.



This is inside view of **Subsystem inside reset bolck. Trigger block** is used to trigger the signal for reset. The parameters for Trigger is shown in beside diagram.

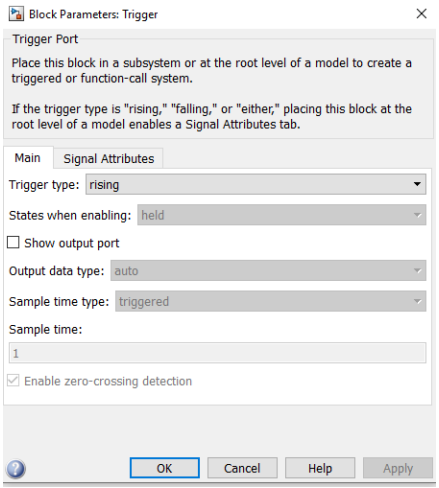
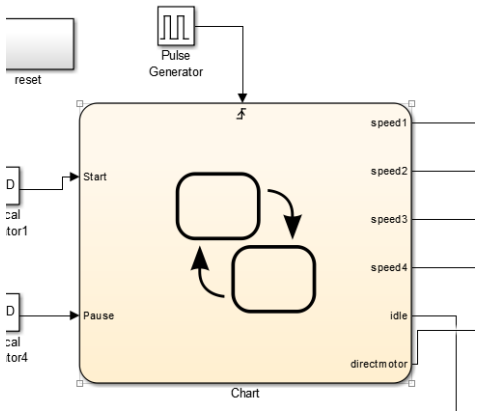
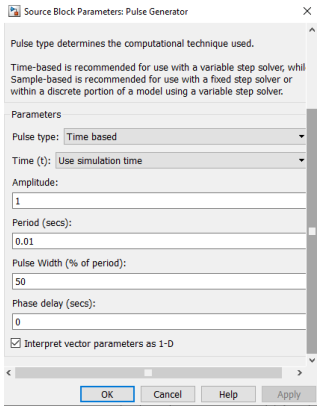


Chart Block

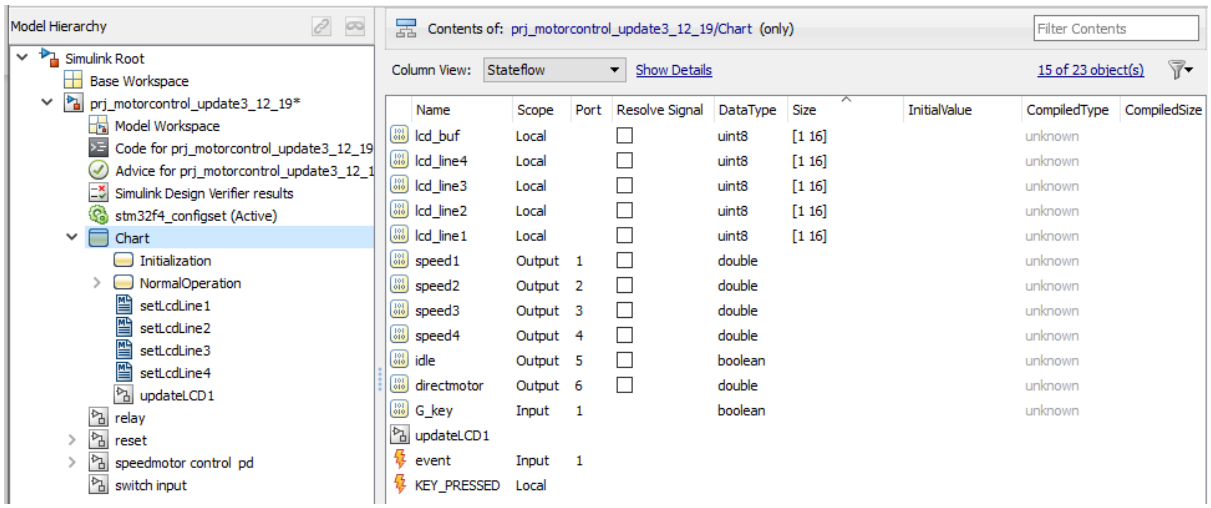


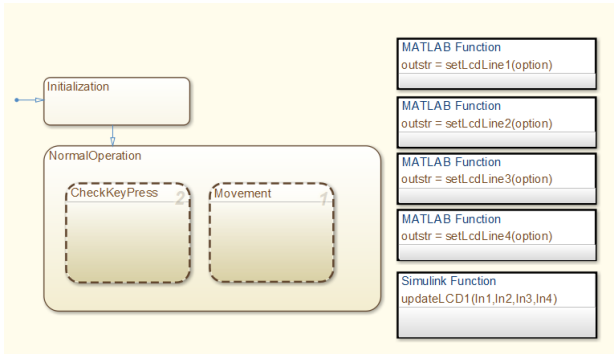
The pulse generator is used for trigger of the **chart**.



This is Block Parameters for **Pulse Generator**

For Model Explorer of Chart





This is inside view of the chart. And these MATLAB Functions are used to write about robot status on LCD Screen of the robot. Simulink function is used to display the robot status on LCD screen.

Codes For 1st line on LCD Screen

```
function outstr = setLcdLine1(option)

switch option
    case 0              %0123456789ABCDEF
        outstr = uint8(' AUTO MOVING CAR');
    otherwise
        outstr = uint8('                ');
end
```

Codes For 2nd line on LCD Screen

```
function outstr = setLcdLine2(option)

switch option
    case 0              %0123456789ABCDEF
        outstr = uint8('      ABAC      ');
    case 1
        outstr = uint8('    *FORWARD*   ');
    case 2
        outstr = uint8('   *BACKWARD*   ');
    case 3
        outstr = uint8('  *MOVE RIGHT*  ');
    case 4
        outstr = uint8('   *MOVE LEFT*  ');
    case 5
        outstr = uint8(' *ROTATING CCW* ');
    case 6
        outstr = uint8(' *ROTATING CW*  ');


    otherwise
        outstr = uint8('                ');
end
```

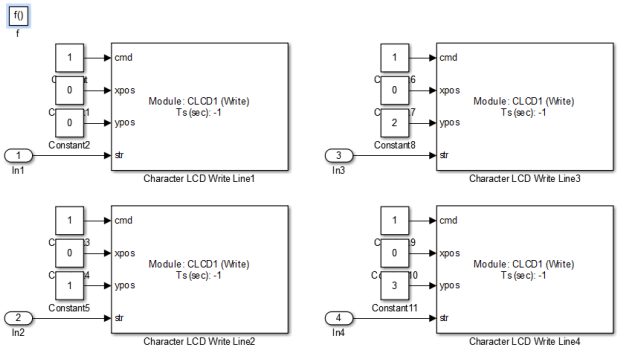
Codes For 3rd line on LCD Screen

```
function outstr = setLcdLine3(option)

switch option
    case 0              %0123456789ABCDEF
        outstr = uint8('  *USER INPUT*  ');
    case 1
        outstr = uint8('                ');
    otherwise
        outstr = uint8('                ');
end
```

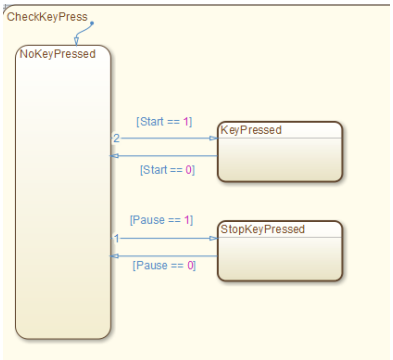
Codes For 4th line on LCD Screen

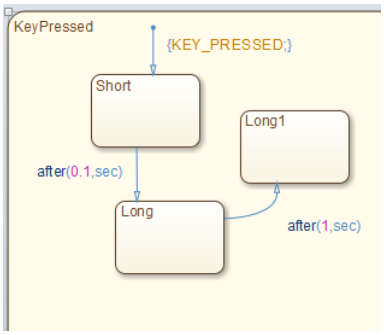
```
function outstr = setLcdLine4(option)

switch option
    case 0              %0123456789ABCDEF
        outstr = uint8('                ');
    case 1
        outstr = uint8('                ');
    otherwise
        outstr = uint8('                ');
end
```

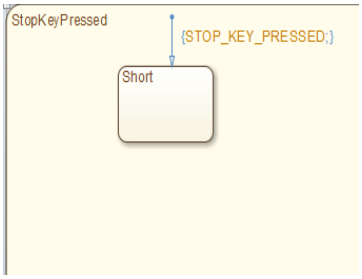
Blocks inside Simulink function in the chart



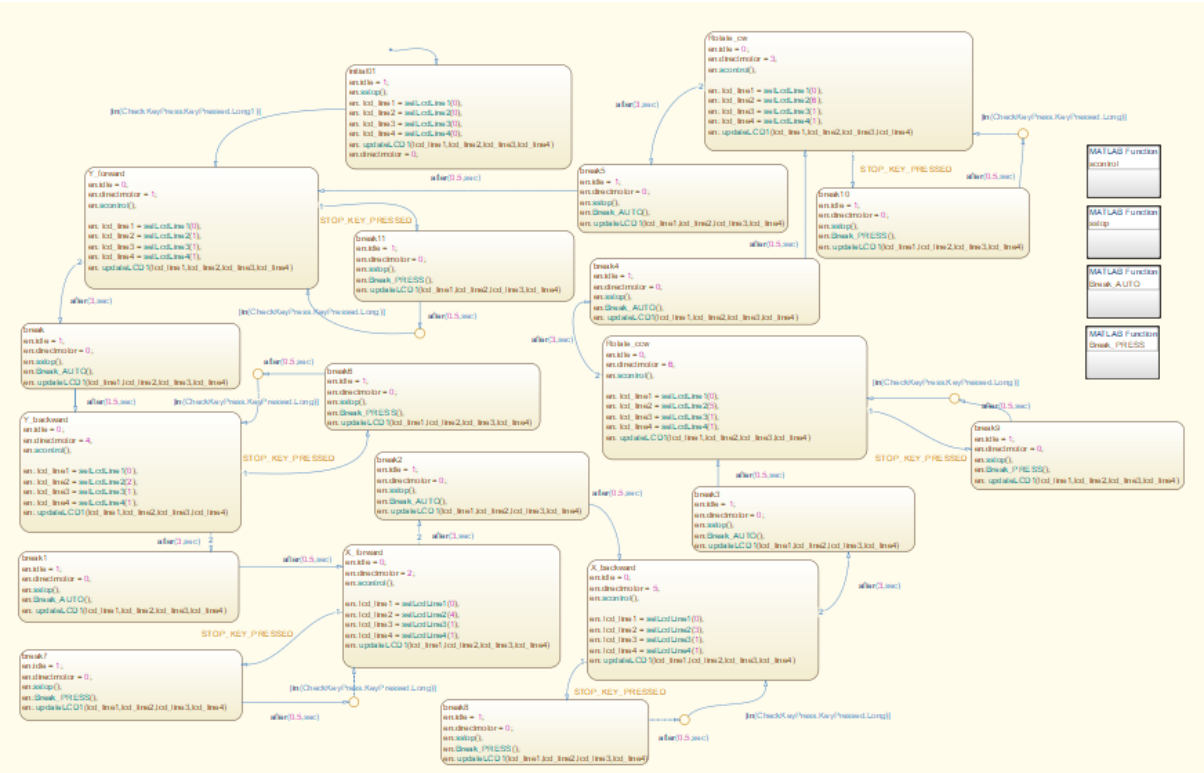
Blocks to make the input Keys operate







Overall block inside Moment Block of Chart



Overall view cannot be seen clearly, so we show the blocks clearly by dividing two parts of this overall block below.

We have 7 main states blocks; each block indicates different directions which are initial state (not moving), Y forward (moving forward), Y backward (moving backward), X forward (moving Right), X backward (moving Left), Rotate CW (Rotating Clockwise direction), Rotate CCW (Rotating Counter Clockwise direction).

In each main states block:
when idle = 0, robot is moving, otherwise, robot is stop.

when directmotor = 1, robot will move Forward.
when directmotor = 4, robot will move Backward.

when directmotor = 2, robot will move Right.
when directmotor = 5, robot will move Left.

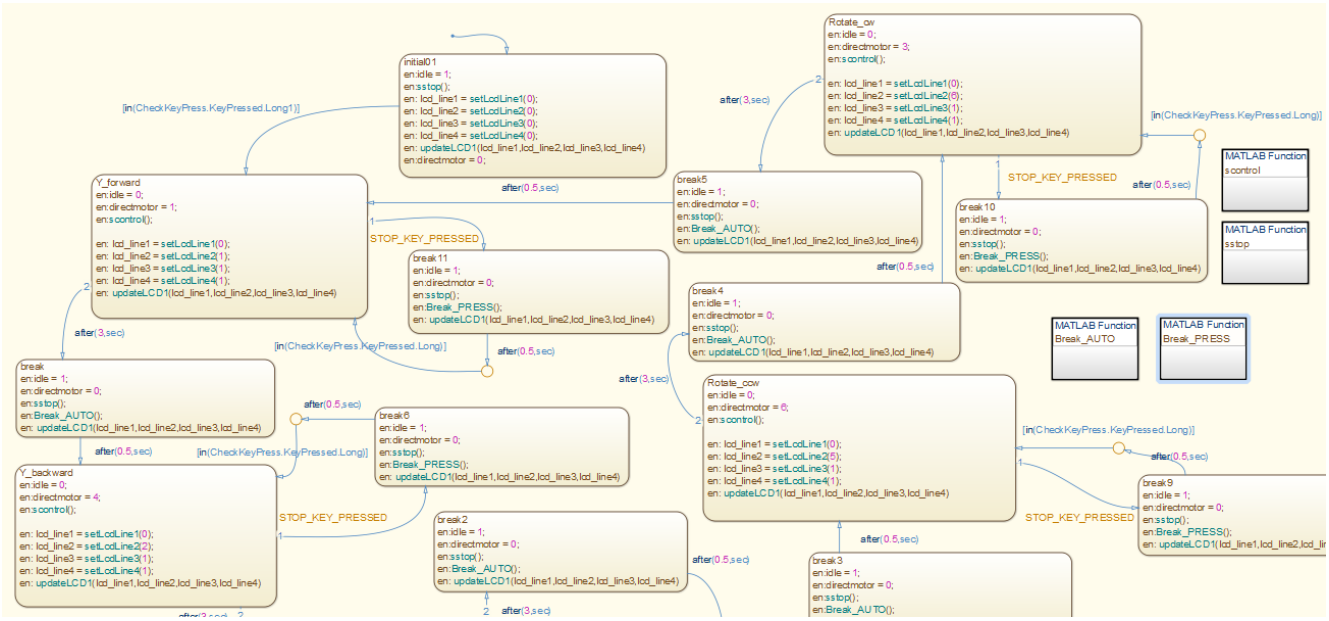
when directmotor = 3, robot will rotate CW.
when directmotor = 6, robot will rotate CCW

For each moving and rotating states, robot will operate for 3 seconds, and then take 0.5 seconds break before changing to next state.

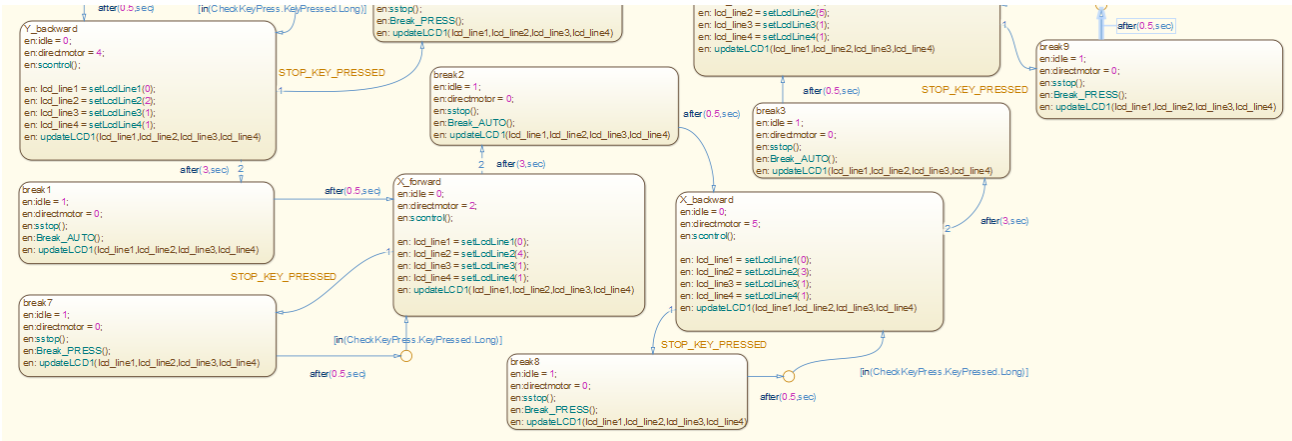
If we pressed start button while robot is moving, it will be paused (stop). And when we press the start button again after 0.5 seconds, it will move or rotate again (in current direction) but the timer is reset, so it will move or rotate 3 seconds. Instead of using "KEY_PRESSED" for start input, we used "[in(CheckKeyPress.KeyPressed.Long)]" and "[in(CheckKeyPress.KeyPressed.Long1)]" to control the pressing time. For stop input, we use "STOP_KEY_PRESSED" because we don't need to control pressing time of stop button.

MatlabFunction blocks are used to control the speed of robot and to change the robot status on LCD screen when it is paused.

Blocks of Upper Part inside Moment Block of Chart



Blocks of Lower Part inside Moment Block of Chart



Codes for Speed Control of scontrol function block

scontrol* | sstop | Break_AUTO | Break_PRESS

```
function scontrol
speed1 = 5;
speed2 = 5;
speed3 = 5;
speed4 = 5;
```

Codes for Speed Control of sstop function block

scontrol* | sstop | Break_AUTO | Break_PRESS

```
function sstop
speed1 = 0;
speed2 = 0;
speed3 = 0;
speed4 = 0;
```

Codes for Speed Control of Break_AUTO function block

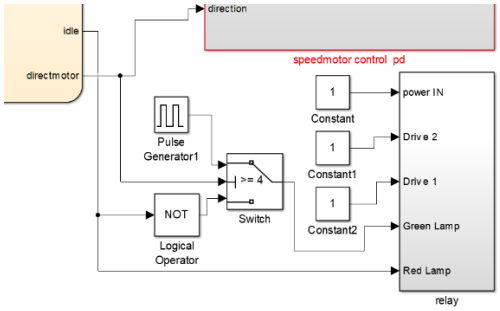
scontrol* | sstop | Break_AUTO | Break_PRESS

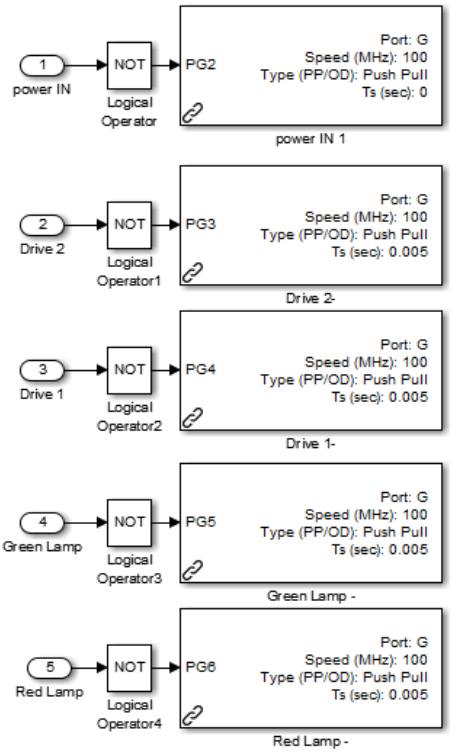
```
function Break_AUTO
coder.ceval('sprintf', coder.ref(lcd_buf),['     BREAK     ' char(0)]);
lcd_line1 = lcd_buf;

coder.ceval('sprintf', coder.ref(lcd_buf),['               ' char(0)]);
lcd_line2 = lcd_buf;

coder.ceval('sprintf', coder.ref(lcd_buf),['  PROGRESSING  ' char(0)]);
lcd_line3 = lcd_buf;

coder.ceval('sprintf', coder.ref(lcd_buf),['               ' char(0)]);
lcd_line4 = lcd_buf;
```

Codes for Speed Control of Break_PRESS function block

scontrol* | sstop | Break_AUTO | Break_PRESS

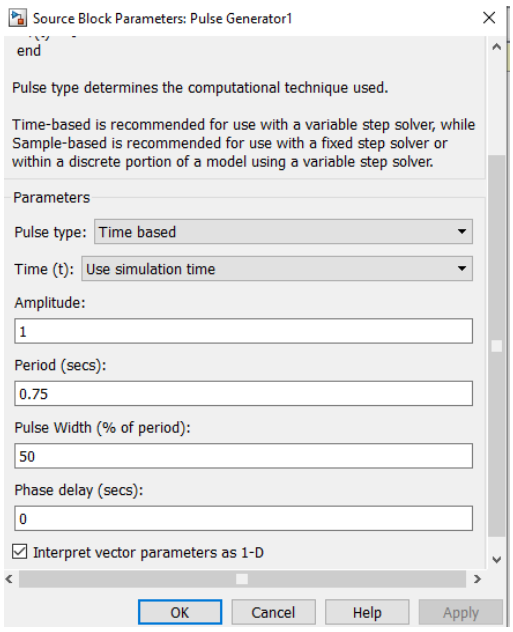
```
function Break_PRESS
coder.ceval('sprintf', coder.ref(lcd_buf),['     BREAK     ' char(0)]);
lcd_line1 = lcd_buf;

coder.ceval('sprintf', coder.ref(lcd_buf),['               ' char(0)]);
lcd_line2 = lcd_buf;

coder.ceval('sprintf', coder.ref(lcd_buf),['PRESS TO RESUME' char(0)]);
lcd_line3 = lcd_buf;

coder.ceval('sprintf', coder.ref(lcd_buf),['               ' char(0)]);
lcd_line4 = lcd_buf;
```

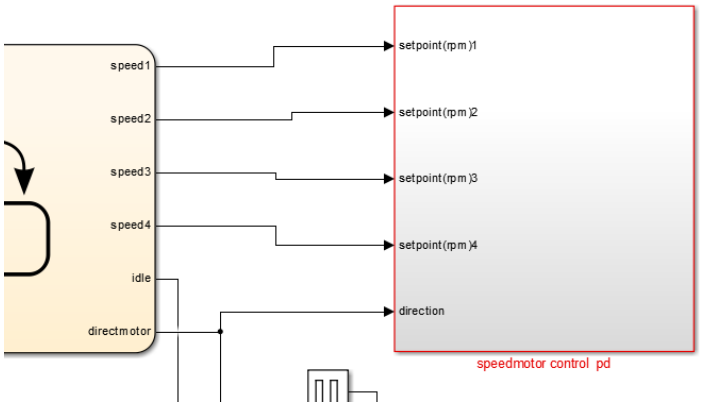




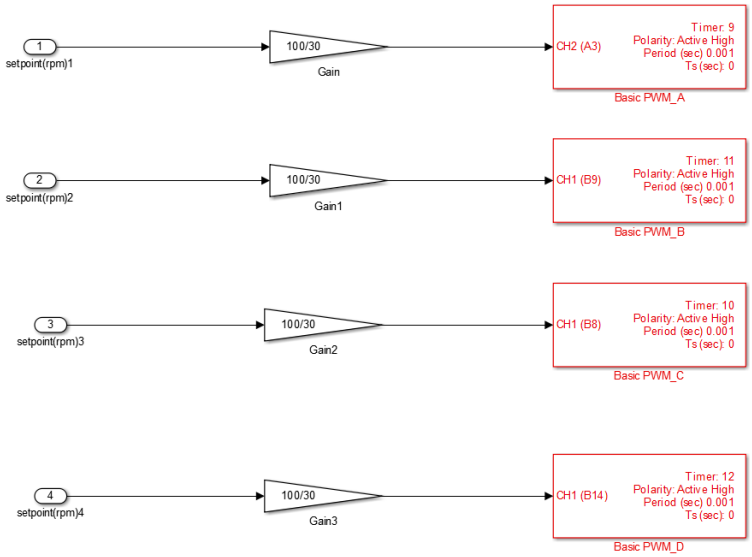
The output (**idle**) of the chart is connected to input (**Red Lamp**) of the relay block. When signal from (idle) is 1, Red Lamp will be ON.
When the output (**directmotor**) signal of the chart is greater than or equal to 4, input (**Green Lamp**) of the relay block will receive the signal from the **Pulse Generator 1**. Otherwise, the **Green Lamp** will receive the signal from **idle**.
The **Constant blocks** are used to give the signal to power IN, Drive 2 and Drive 1.

Block Parameters for Pulse Generator 1

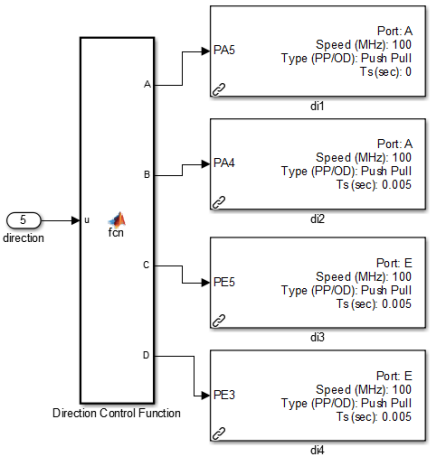




The output (**speed 1- 4**) of the chart is connected to the input (**setpoint (rpm) 1-4**) of the **speedmotor control pd block**. The output (**directmotor**) of the chart is connected to the input (**direction**) of the speedmotor control pd block.



The signals from the **setpoints** (1- 4) are used to set the speed of the motor. Gain (0-3) are used to amplify the signals from setpoints (1- 4).



The signal from the direction (5) is fed into the **Direction Control function block**. Inside it, conditions are set to case 1 through 5, and an offset condition (otherwise). This MATLAB function is used to control the direction of the motors.

speedmotor control pd/MATLAB Function

```
function [A,B,C,D]= fcn(u)
switch u
    case 1 %y forward
        A = 1;
        B = 1;
        C = 0;
        D = 0;
    case 4 %y backward
        A = 0;
        B = 0;
        C = 1;
        D = 1;
    case 2 %x forward
        A = 0;
        B = 1;
        C = 1;
        D = 0;
    case 5 %x backward
        A = 1;
        B = 0;
        C = 0;
        D = 1;
    case 3 %cw
        A = 1;
        B = 1;
        C = 1;
        D = 1;
    otherwise %ccw
        A = 0;
        B = 0;
        C = 0;
        D = 0;
end
end
```

Codes for Direction Control Function Block

IV. Results

1. Result of directional control of the mobile robot is shown in videos.
2. Result of LCD screen display are shown in below and in video too.

Stage 1: Initial Stage

Stage 2: Move Forward

Stage 3: Move Backward

Stage 4: Move Left

Stage 5: Move Rights

Stage 6: Rotate CCW

Stage 7: Rotate CW

Break Stage 1:
After each stages is run for 3 seconds, mobile will take a break (stop) for 0.5 second before changing to next state.

Break Stage 2:
When we press stop button, mobile robot will stop until we press start button.

V. Discussion and Conclusion

For the project, we studied about the start-up source codes of the mobile robot (given by our lecturer) and about directional control of the drives, and about the pin connections (especially the motor drive pins) on microcontroller.

In this project, we could sucessfully control the direction of this mobile robot by dividing the stages (for directions) and set the time for each stages. In testing video, this mobile robot sometimes doesn't go straight due to the slippy floor; the wheels are slipped with smooth surface of the floor and the direction changes a bit. Therefore, we recommend to test this robot in not very smooth floor to make it move in correct direction.

For further development of this project, if we want to contol this mobile robot by a controller (using mobile application), we can add serval inputs in chart. And we can use the press key buttons (example, PRESS_FORWARD, PRESS_BACKWARD, etc.) instead of setting the time for each stage. Moreover, we have to use the knowledge of connecting between ESP8266 (wifi module) and simulink.

To conclude, we could apply our knowledge from this Microprecessor class in this project. And we gained more understanding about the function and process of Simulink and Waijung Boxset. We gained more knowledge about MATLAB functions and Microprocessor system, and also directional control of the motor drive.