CmpE 443 Final Project Design Document

Group Name : Araba Sevdası

Members
Atıf Emre Yüksel (Team Leader)
Cemal Burak Aygün
Ömer Kırbıyık
Yaşar Alim Türkmen

December 29, 2018
Version 2.0

Contents

1 System Level Structural Diagram (Block Diagram)

This is the system level structural diagram of the car we designed.

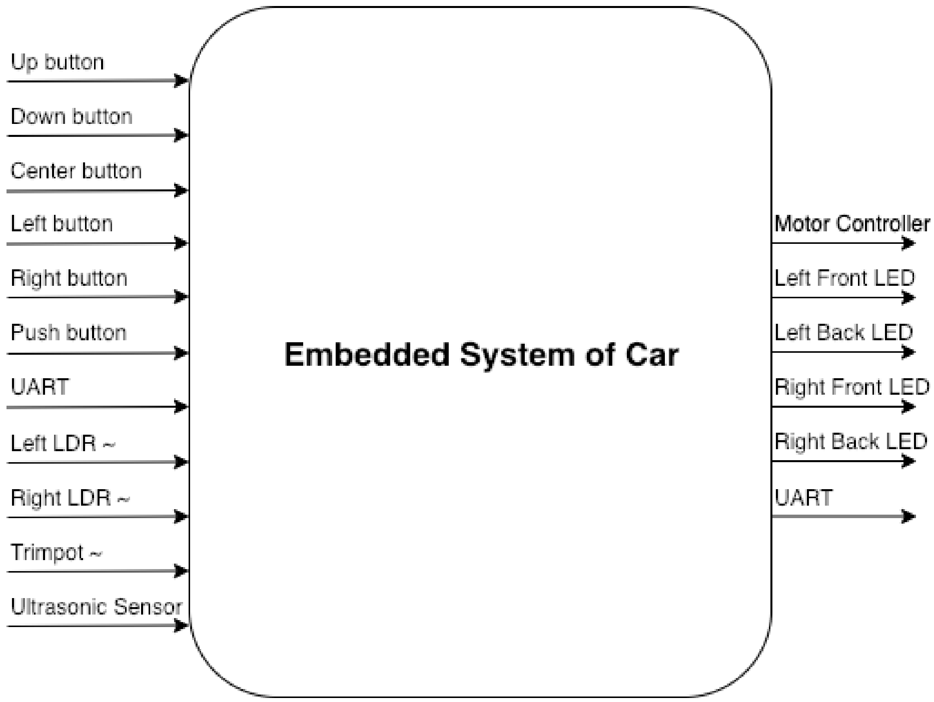

Figure 1: System Level Structural Diagram

Push button is used to toggle the mode of the car between MANUAL and AUTO. Joystick buttons (up, down, left, right and center) on the base board are used as the input which set the moving behaviour of the car. In MANUAL mode, those behaviors are moving forward, moving backward, counter clockwise point rotation, clockwise point rotation, and stopping, respectively. In AUTO mode, only the UP joystick button is effective and it makes the car start moving. Mode of the car can also be changed via UART. When '*' is sent, the mode is set to MANUAL and when '#' is sent, the mode is set to AUTO. Also, UART is used as a start signal in AUTO mode. When "66" is sent in AUTO mode, the car starts moving.

There are two light dependent resistors on the car, one is on the front-left and one is on the front-right side. Those LDRs detect the light change and decide the car's left-right turning (rotating) move when it is going forward. Also, there is an ultrasonic sensor mounted on the front side of the car. It is used to detect obstacles and avoid crashing into them while the car is moving forward. The trimpot adjusts the speed of the car.

In the output part, we have four LEDs, one Motor Controller and UART. The controller is used to perform the moving behaviors mentioned above. The LEDs are used to show the direction in which the car goes basically. When the car goes forward, front LEDs are lighted. When the car goes backward, back LEDs are lighted. When the car rotates left (i.e. counter clockwise), left LEDs are blinked. When the car rotates right (i.e. clockwise), right LEDs are blinked. Finally, when the car stops, all the LEDs are turned off. UART is used to display the current mode of the car. When the mode changes, "MANUAL" or "AUTO" is written to UART.

When the system is started, the car is in stopped state. At this point, PWM0, Timer2 (used for ultrasonic sensor operation), Timer3 (used for LED blinking), UART0, ADC (used for trimpot and LDRs), push button and joystick buttons are initialized. Timer2 and Timer3 modules are disabled on initialization. Other components remain enabled as long as the system runs. Timer2 is enabled when the car moves forward. When the car starts moving another direction or stops, it is disabled again. Timer3 is enabled only for left or right rotation actions. When the car stops, it is disabled again. Mode change causes the car to stop.

In the logic of algorithm, the car decides which direction it goes by using the output of the light dependent resistor (LDR) values. The raw values of LDRs are converted to numbers between 0 and 100. As the intensity of the light increases, the value gets greater. At first, the ultrasonic sensor value is checked to control whether the car encounters an obstacle or not. If it encounters an obstacle like another car closer than 15 cm, the car starts to move backward until 30 cm distance is reached from the obstacle. Therefore, ultrasonic sensor has the priority over LDRs in movement decision while going forward. If the car does not perceive an obstacle and the left LDR's value is higher than some threshold, the car starts point rotation to turn left until it reads a left LDR value that is lower than its threshold. The same process works for the right LDR to turn right.

2 Sequence Diagrams

2.1 Moving Car Forward

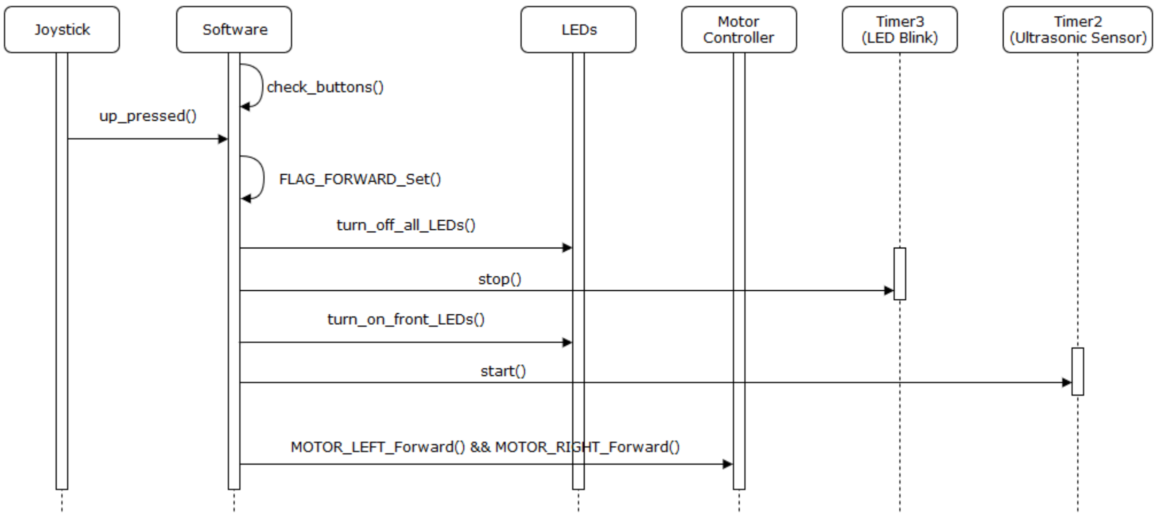

Figure 2: Moving Car Forward (Sequence Diagram)

2.2 Moving Car Backward

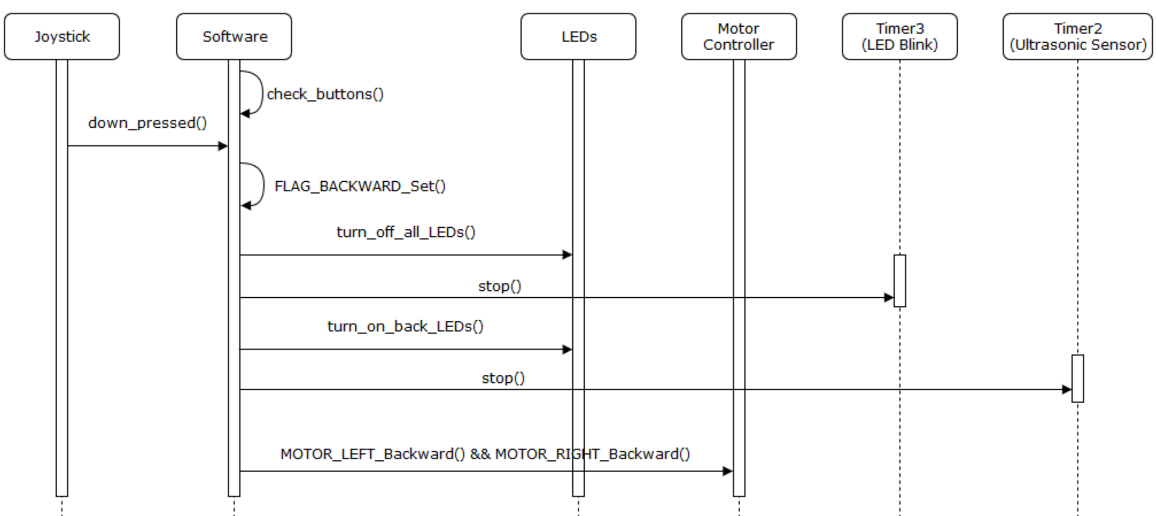

Figure 3: Moving Car Backward (Sequence Diagram)

2.3 Rotating Car to Left

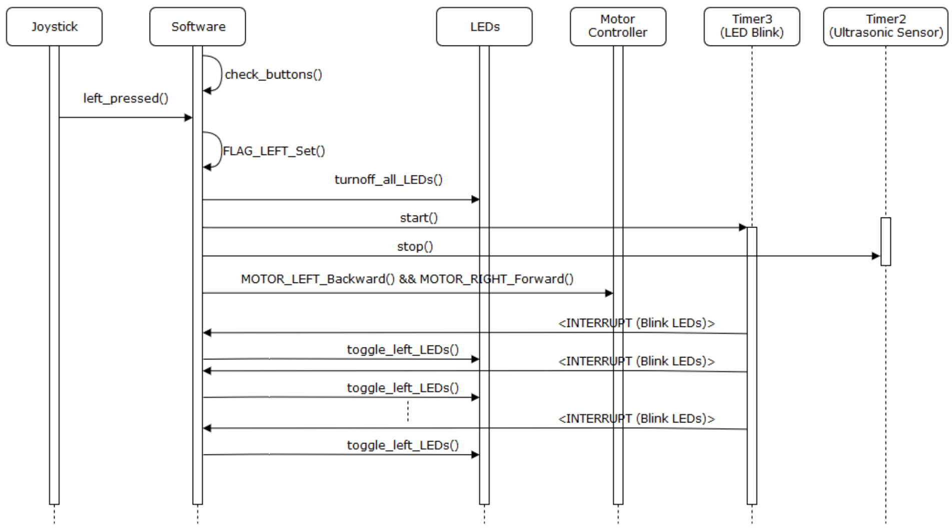

Figure 4: Rotating Car to Left (counter clockwise) (Sequence Diagram)

2.4 Rotating Car to Right

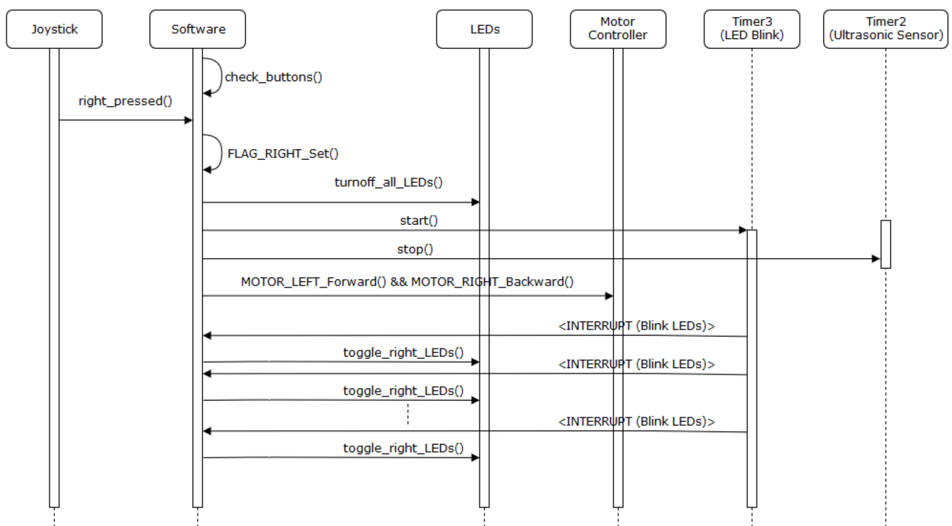

Figure 5: Rotating Car to Right (clockwise) (Sequence Diagram)

2.5 Stopping Car

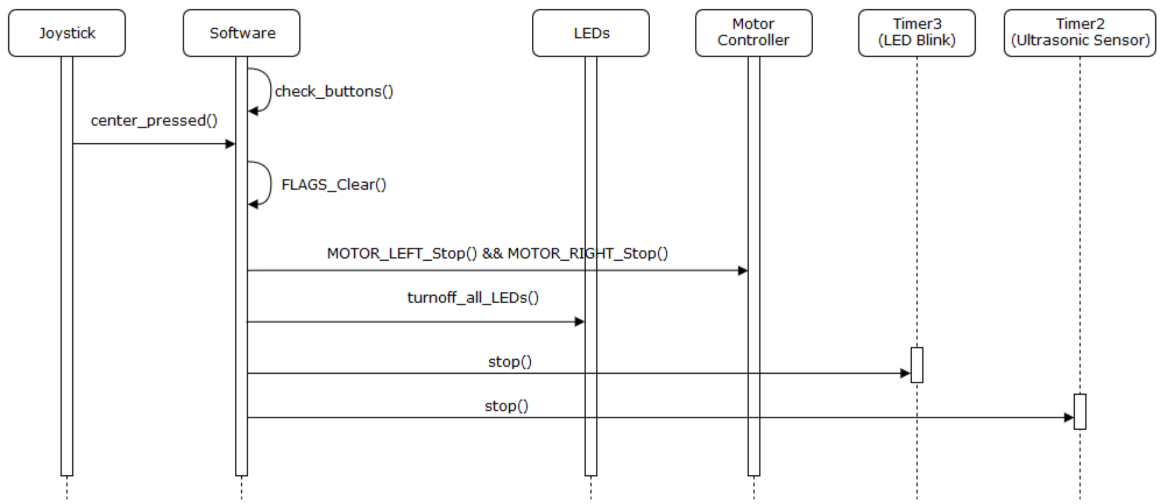

Figure 6: Stopping Car (Sequence Diagram)

2.6 Changing Mode via UART

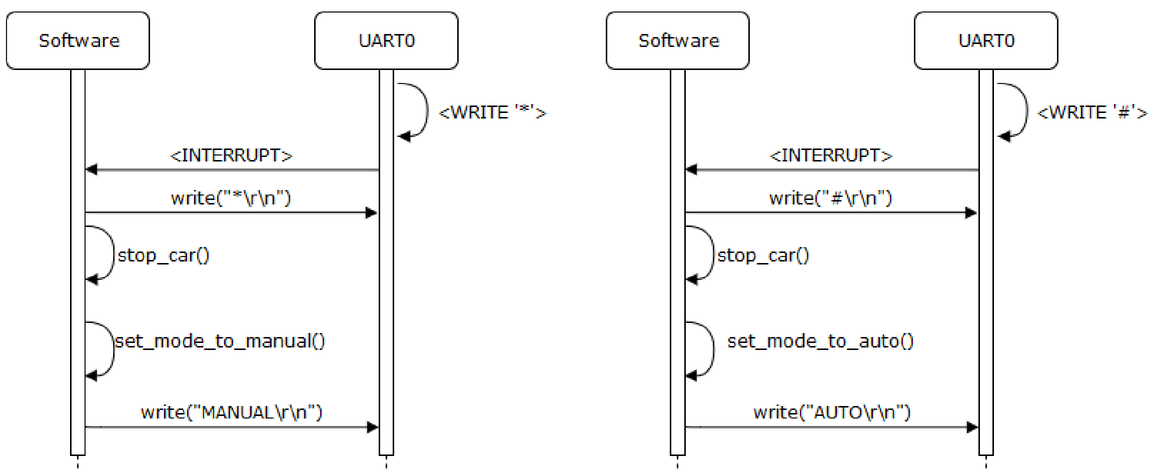

Figure 7: Changing Mode via UART (Sequence Diagram)

2.7 Changing Mode via Push Button

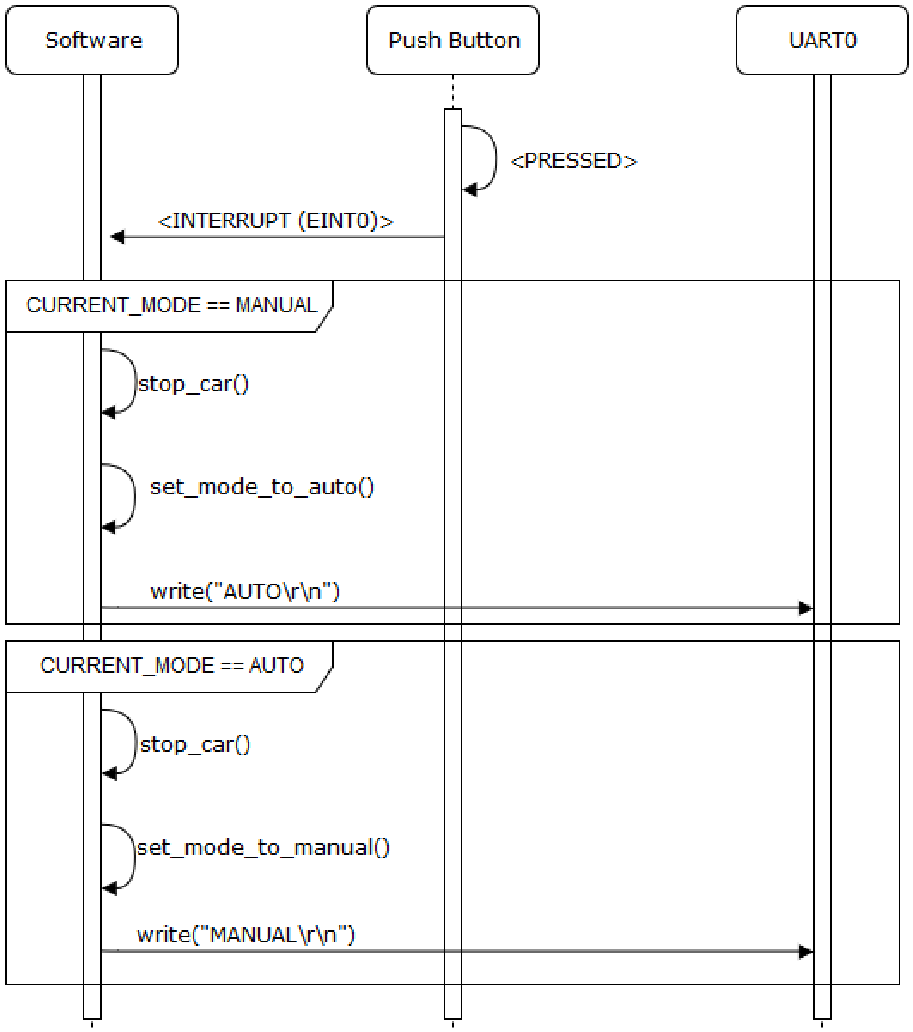

Figure 8: Changing Mode via Push Button (Sequence Diagram)

2.8 Capturing Distance via Ultrasonic

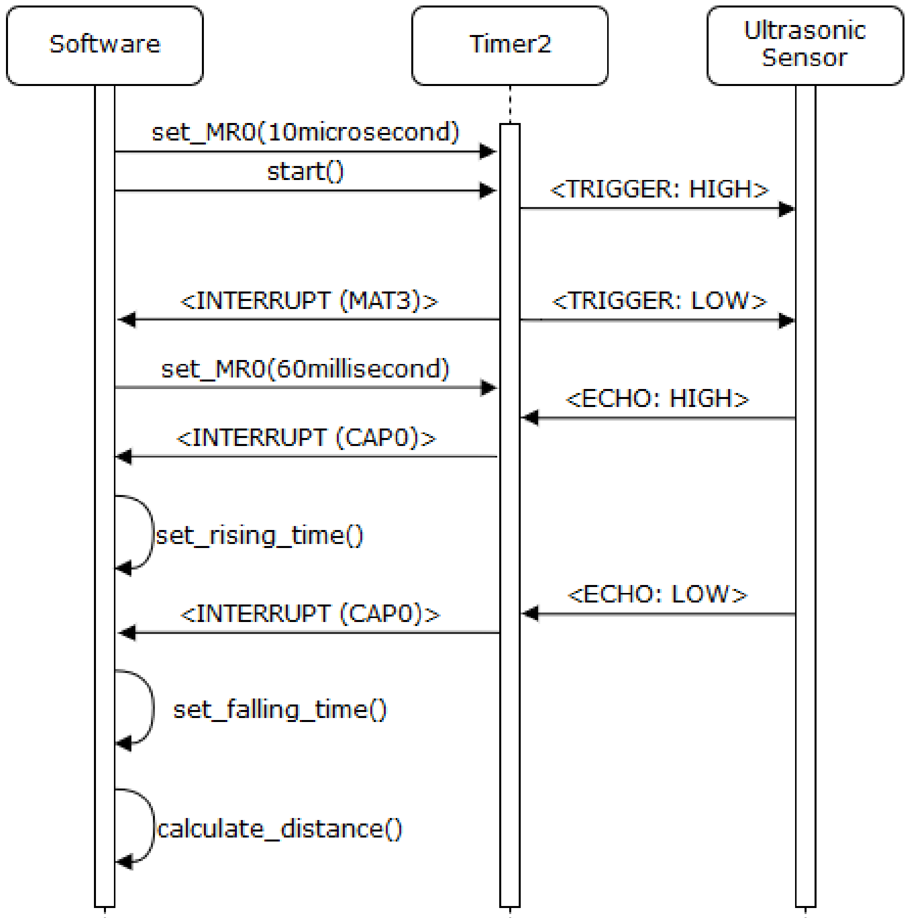

Figure 9: Capturing Distance via Ultrasonic (Sequence Diagram)

2.9 Getting Trimpot Value

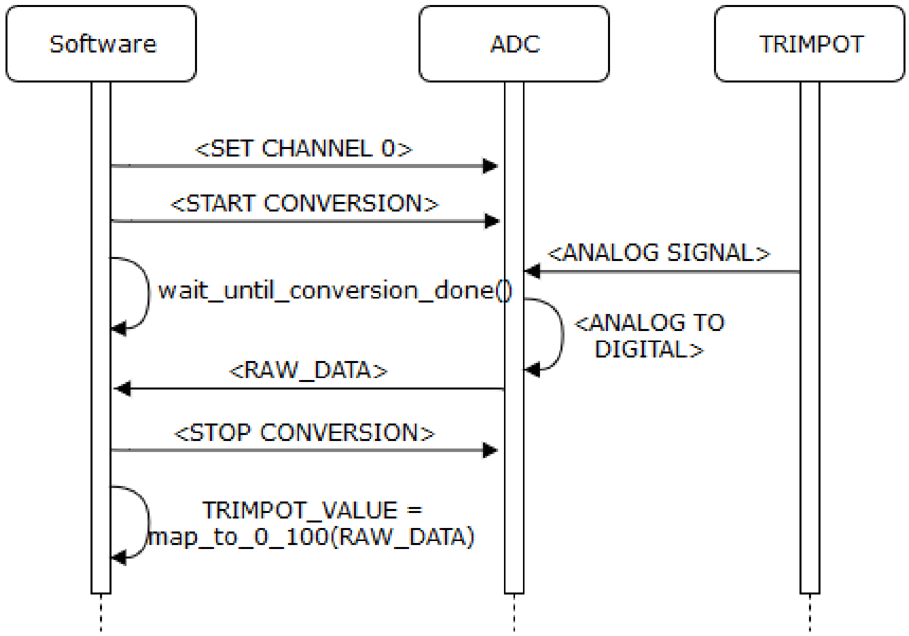

Figure 10: Getting Trimpot Value (Sequence Diagram)

2.10 Getting LDR Values

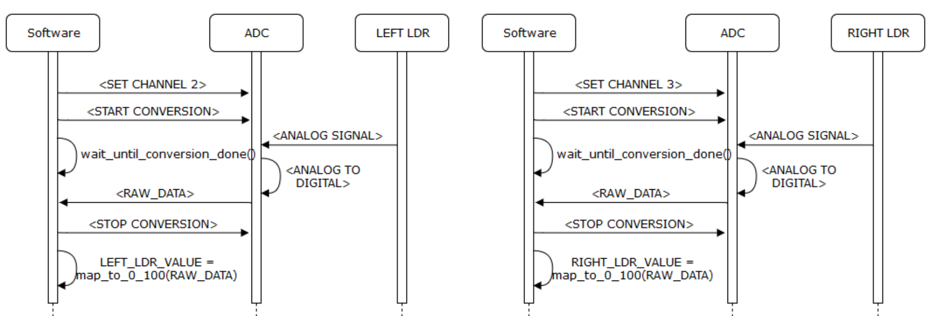

Figure 11: Getting LDR Values (Sequence Diagram)

3 main() Function Pseudocode

```
initCompoenents()

leftLDRThreshold ← 40
rightLDRThreshold ← 50
escapingFromLight ← False

while True do
   currentTrimpot ← TRIMPOT_Read()
   CAR_SetSpeed(currentTrimpot, currentTrimpot)

   if GOING_FORWARD then
      if DISTANCE_TO_OBSTACLE ≤ 15 then
         CAR_Backward()
         ULTRASONIC_Start()
      else
         leftLDR ← LDR_LEFT_Read()
         rightLDR ← LDR_RIGHT_Read()

         if (leftLDR > leftLDRThreshold) or (rightLDRThreshold-30 > rightLDR and
         rightLDR < rightLDRThreshold-20) then
            CAR_RotateRight()
            escapingFromLight ← True
         else if (rightLDR > rightLDRThreshold) or (leftLDRThreshold-30 < leftLDR
         and leftLDR < leftLDRThreshold-20) then
            CAR_RotateLeft()
            escapingFromLight ← True
         end if
      end if

   else if GOING_BACKWARD then
      if ESCAPING_FROM_OBSTACLE and DISTANCE_TO_OBSTACLE ≥ 30 then

         CAR_Forward()
      end if

   else if ROTATING_LEFT then
      if ESCAPING_FROM_LIGHT then
         rightLDR ← LDR_RIGHT_Read()
         if rightLDR ≤ rightLDRThreshold then
            escapingFromLight ← False
            CAR_Forward()
         end if
      end if
```

```
  else if ROTATING_RIGHT then
    if ESCAPING_FROM_LIGHT then
      leftLDR ← LDR_LEFT_Read()
      if leftLDR ≤ leftLDRThreshold then
        escapingFromLight ← False
        CAR_Forward()
      end if
    end if
  end if

  if CURRENT_MODE == MODE_MANUAL then
    if JOYSTICK_LEFT_Pressed() then
      CAR_RotateLeft()
    else if JOYSTICK_RIGHT_Pressed() then
      CAR_RotateRight()
    else if JOYSTICK_UP_Pressed() then
      CAR_Forward()
      JOYSTICK_DOWN_Pressed()
      CAR_Backward()
    else if JOYSTICK_CENTER_Pressed() then
      CAR_Stop()
    end if
  else
    if JOYSTICK_UP_Pressed() then
      CAR_Forward()
    end if
  end if
end while
```

4 LED Connections

LED	LPC4088 PIN	FUNCTION
Front-Left	P0_0 (P9)	GPIO
Front-Right	P0_1 (P10)	GPIO
Back-Left	P0_8 (P12)	GPIO
Back-Right	P0_7 (P13)	GPIO

Table 1: LED Connections

REASON for the Selected PINS: We need 3 states for the LEDs: On, off and blink. For the blink action, our first idea was to use PWM. However, base board provide us with one PWM module which is PWM0. We use PWM0 to operate Motor Controller. Since the periods of the LEDs and the motors are different, we couldn't proceed with this idea.
Our second idea was to use timer external match pins and make those pins toggle periodically on TC matching MR0. Only Timer2 provides 4 external match pins (P11 to P14) on base board. However, P14 does not work properly because the base board uses it internally for something else. Hence, we left this idea, too.
Finally, we decided to connect our LEDs to GPIO pins and use a timer (Timer3) to generate periodic interrupts in which we toggle the value (from HIGH to LOW or from LOW to HIGH) of the related pins to make the LEDs blink.
After we assigned pins on the board to other components which require special functionalities (such as timer match, timer capture, analog-to-digital conversion, etc.), we selected those 4 pins among the remaining ones for LEDs. We wanted the four pins to be on the same port to make easy for GPIO read-write operations.

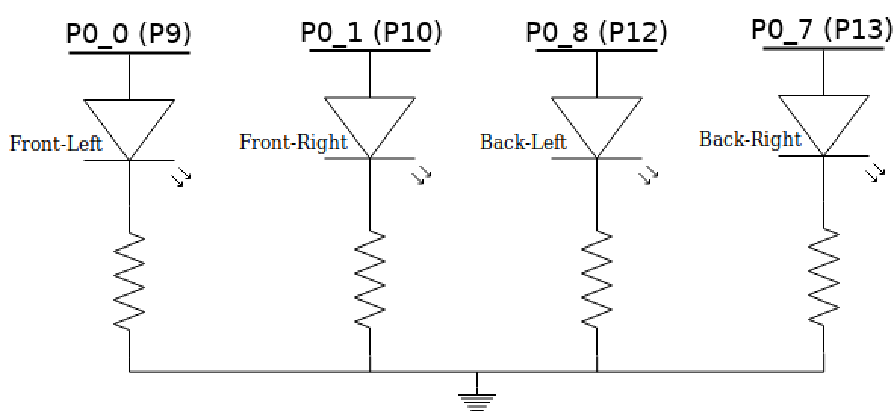

Figure 12: LED Circuit

5 Motors - Motor Driver Connections

MOTOR TERMINAL	MOTOR DRIVER TERMINAL
Front-Left Motor +	**Out1**
Front-Left Motor -	**Out2**
Front-Right Motor +	**Out4**
Front-Right Motor -	**Out3**
Back-Left Motor +	**Out1**
Back-Left Motor -	**Out2**
Back-Right Motor +	**Out4**
Back-Right Motor -	**Out3**

Table 2: Motors - Motor Driver Connections

We have 4 motors and only one Motor Controller which has 2 motor outputs. Hence, we have to connect 2 motors to 1 output of the controller. Our car should have the ability to rotate clockwise and counter clockwise. This means that the left motors and the right motors could run in opposite directions at the same time. As a result, we connected the left motors to one output of the controller and the right motors to the other output.

We connected the motors to the controller such that:

- Left motors are on OutputA (Out1 & Out2) and right motors are on OutputB (Out3 & Out4)
- When IN1 is LOW and IN2 is HIGH, left motor runs in forward direction, when IN1 is HIGH and IN2 is LOW, left motor runs in backward direction
- When IN3 is LOW and IN4 is HIGH, right motor runs in forward direction, when IN3 is HIGH and IN4 is LOW, right motor runs in backward direction

6 Motor Driver - Board Connection

DRIVER PIN	LPC4088 PIN	FUNCTION
ENA	**P1_2 (P30)**	**PWM0[1]**
ENB	**P1_3 (P29)**	**PWM0[2]**
IN1	**P1_5 (P28)**	**GPIO**
IN2	**P1_6 (P27)**	**GPIO**
IN3	**P1_7 (P26)**	**GPIO**
IN4	**P1_11 (P25)**	**GPIO**

Table 3: Motor Driver - Board Connection

REASON for the Selected PINs: Motor Controller has two motor outputs and each of them needs 1 pin with PWM to set its speed and 2 GPIO pins to set its direction. Base board provides us with one PWM module which is PWM0. PWM0 has six channels. We connected the first two channels of PWM0 (P30 and P29) to ENA and ENB pins of the controller. Since, other 4 channels of the PWM0 are not used for any other components in the system and we wanted to group (physically) all of the pins related to the controller together, we used P28-P25 in GPIO mode and connected those pins to the direction pins of the controller.

7 Ultrasonic Sensor - Board Connection

ULTRASONIC SENSOR PIN	LPC4088 PIN	FUNCTION
VCC	**VU**	**VU**
GND	**GND**	**GND**
TRIG	**P0_9 (P11)**	**T2_MAT3**
ECHO	**P0_4 (P34)**	**T2_CAP0**

Table 4: Ultrasonic Sensor - Board Connection

REASON for the Selected PINs: Ultrasonic sensor requires a signal of square wave with (at least) 60ms period and 10 microseconds on-time. One way to produce this signal is to use PWM. However, only PWM module (PWM0) available on the board is utilized to operate the motors. Another way to generate this signal is to utilize a Timer and use one of its output match pins. We used MAT3 pin of Timer2 for this purpose. Using the interrupt mechanism, Timer2 gives its MAT3 pin HIGH for 10 microseconds and LOW for 60ms.

When the ultrasonic sensor is triggered, it sets its ECHO pin to HIGH. When the measurement of a distance is completed, ECHO pin is set to LOW. We need to gather the time passed between the rising and falling edges of the ECHO pin, which can be done using a timer and one of its input capture pins. We are already using a timer (Timer2) for triggering the ultrasonic sensor. So, we used the same timer also for capturing the ECHO pin. Calculating the distance is also done via interrupt mechanism.

8 Light Dependant Resistor (LDR) Connections

LDR PIN	LPC4088 PIN	FUNCTION
Left-LDR	**P0_25 (P17)**	**ADC0[2]**
Right-LDR	**P0_26 (P18)**	**ADC0[3]**

Table 5: Light Dependant Resistor (LDR) Connections

REASON for the Selected PINs: LDR is light-controlled variable resistor. As the incident light intensity increases, the resistance decreases. We use the voltage value on the LDR as input, which is analog. Hence, we need to connect LDR pin to one of the pins on the board with ADC (analog-to-digital convertion) functionality.
There are 6 pins on our board (pins P15 to P20) that have the ADC functionality. The Trimpot on the board is connected to P15 and we use the Trimpot as well. So, we selected those 2 pins among the remaining ones for LDRs.
We constructed LDR circuits such that as the light intensity on the LDR increases, the input value of the pin also increases.

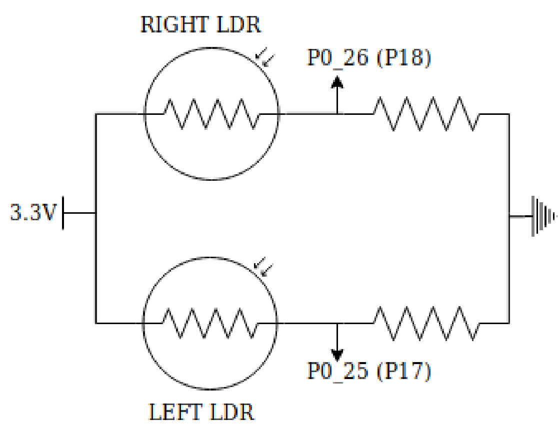

Figure 13: LDR Circuit