CmpE 443 Final Project Report

Group Name : R3S3RV3D

Members

Dilruba Reyyan KILIÇ (Team Leader)

Yusuf KALAYCI

Bekir Burak ASLAN

Mustafa Enes ÇAKIR

December 28, 2018

Version 2.00

R3S3RV3D

Contents

1 Introduction

In this report, we will introduce the progress of our group: R3S3RV3D on the final project of CmpE 443 (Embedded Systems) Course.

At this point, the Car includes 4 LED's, 1 Motor Controller, 2 LDR's, 1 Ultrasonic Sensor and 1 WIFI in addition to the Main Driver LPC4088. The car is able to move on two different modes as *Manual* and *Autonomous*. Manu The car also has an obstacle detector and light sensor which allows it to escape from obstacles and follow a road mapped by lights. The actions and the conditions are explained in detail below.

The challenge is to bring these different devices and controllers together without having any errors or any other complications and create the desired car of the project (Figure 1). It is both challenging at software and hardware design.

In order to explain the hardware connections and software-hardware relations, we prepared this report.

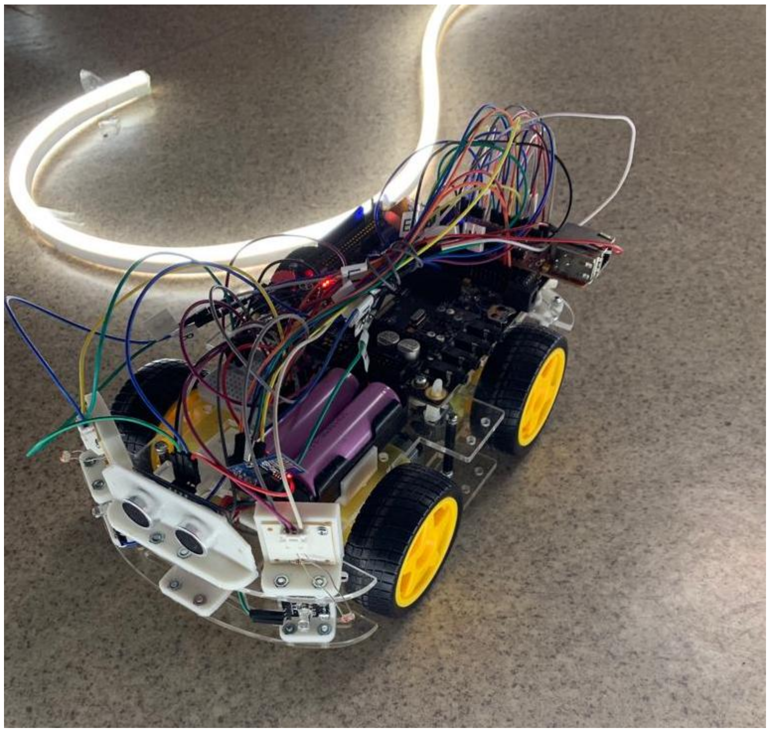

Figure 1: R3S3RV3D Car

2 Block Diagram (System Level Structural Diagram)

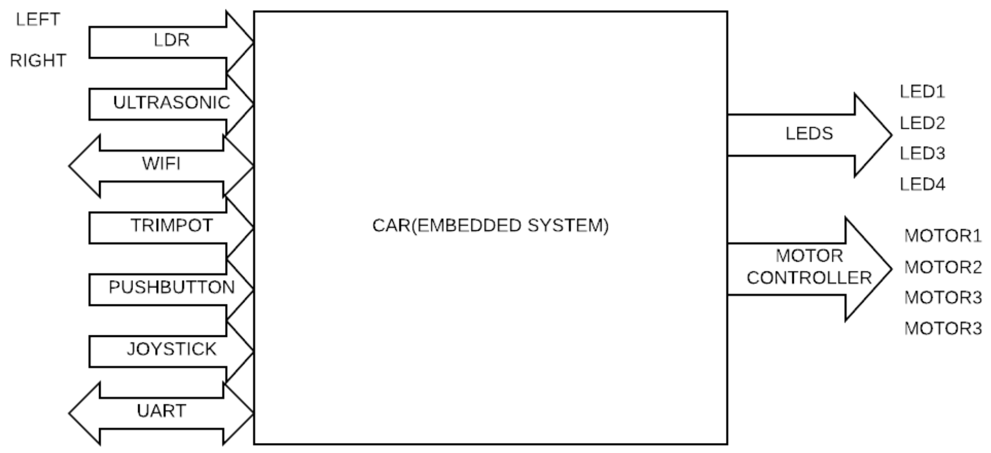

Figure 2: System Level Structural Diagram

2.1 Inputs

2.1.1 Push Button

The push button which is already installed on the board is used to change the mode the car. It switches between the Manual Mode and Autonomous Mode.

2.1.2 Trimpot

The Trimpot which is already installed on the board is used to change the speed of the car. The speed is increased by turning it clockwise.

2.1.3 Joystick

The actions of the car when the Joystick is pressed differs on the Mode of the car. We can list these conditions as:

- When **Joystick Left** button is pressed, the car turns left on the Manual Mode. No action on Autonomous Mode.
- When **Joystick Right** button is pressed, the car turns right on the Manual Mode. No action on Autonomous Mode.
- When **Joystick Up** button is pressed, the car moves forward on the Manual Mode. It activates the Car in Autonomous Mode.

- When **Joystick Down** button is pressed, the car goes in backward direction on the Manual Mode. No action on Autonomous Mode.
- When **Joystick Center** button is pressed, the car stops in both Modes.

The LEDs actions are explained in the LEDs section.

2.1.4 UART/WIFI

The robot(embedded car) switch in terms of autonomous and manual mode. This switch will be done via WiFi or UART. We tried here both of them when we were doing this part. The both of them was similar some ways to the other. Star and diez symbol used for the switching. The symbols means are manual and autonomous respectively. There was a send and receive methodology and for WiFi case SSID:HWLAB and Password:12345678. We successfully change the mode of the car via WiFi. But WiFi communication blocks everything else. The robot can't read values from LDR. UART works properly.

2.1.5 Ultrasonic

The Ultrasonic Sensor is used to detect any obstacle in the range of 15 cm's. When the car detects an obstacle it goes backwards till the obstacle is 30 cm away and move forward again.

2.1.6 Light Sensor (LDR)

There are two LDR's, one on the front-left and another on the front-right of the car. LDR is used to measure the light from the road lines which are lights and determine whether to turn the car or not. We used the value of the difference on the LDR's. The THRESHOLD can be changed via code.

2.2 Outputs

2.2.1 LEDs

The LEDs represents the direction or action of the car. The following states of the LED is changed according to the action which the car performs:

- All LEDs are **OFF**, while the car is not moving.
- The front LEDs are **ON**, while the car is moving forward.
- The back LEDs are **ON**, while the car is moving backward.
- The left LEDs are **BLINKING**, while the car is turning left.
- The right LEDs are **BLINKING**, while the car is turning right.

2.2.2 Motor Controller

The Motor Controller provides the connection between the main driver (LPC4088) and the motors (wheels). The first channel of the Motor Controller controls the left wheels and the second channel of the Motor Controller controls the right wheels. We can obtain the following actions as:

- The car moves FORWARD when IN1 & IN3 are set to 1, and IN2 & IN4 are set to 0.
- The car moves BACKWARD when IN2 & IN4 are set to 1, and IN1 & IN3 are set to 0.
- The car STOPS when IN1 & IN2 & IN3 & IN4 are set to 0.
- The car TURNS RIGHT when IN2 & IN3 are set to 1, and IN1 & IN4 are set to 0.
- The car TURNS LEFT when IN1 & IN4 are set to 1, and IN2 & IN3 are set to 0.

3 Modes

The CAR has two different modes for moving characteristics as *Manual* and *Autonomous* which are explained in detailed below.

3.1 Manual Mode

- The robot is at the stop phase when robot enters manual mode.
- When Joystick Left button is pressed, robot starts to rotate counter-clockwise direction.
- When Joystick Up button is pressed, robot starts to travel in forward direction.
- When Joystick Down button is pressed, robot starts to travel in backward direction.
- When Joystick Center button is pressed, robot stops.
- When Joystick Right button is pressed, robot rotates clockwise direction.
- While robot is going in forward direction, if Ultrasonic sensor detects an obstacle which is 15 cm away from the robot, robot drives in backward direction until ultrasonic sensor reads at least 30 cm. Then, drives in forward direction.
- The robot speed is changable via Trimpot.

- While robot is going in forward direction, if robot detects an light source from the left or right side, the robot escapes from the light source. If light source is in the left side, robot goes to the right side and if light source is in the right side, robot goes to the left side.

3.2 Autonomous Mode

- The robot is at the stop phase when it enters autonomous mode.
- The autonomous execution starts with two different ways: When the Joystick Up is pressed or when '66' is sent via UART or WiFi, robot executes the order and starts moving according to LDR sensor values.
- The robot speed is changable via Trimpot.
- While robot is going in forward direction, if Ultrasonic sensor detects an obstacle which is 15 cm away from the robot, robot drives in backward direction until ultrasonic sensor reads at least 30 cm. Then, drives in forward direction.
- The robot goes along the road.

4 System-Level Functional Diagram

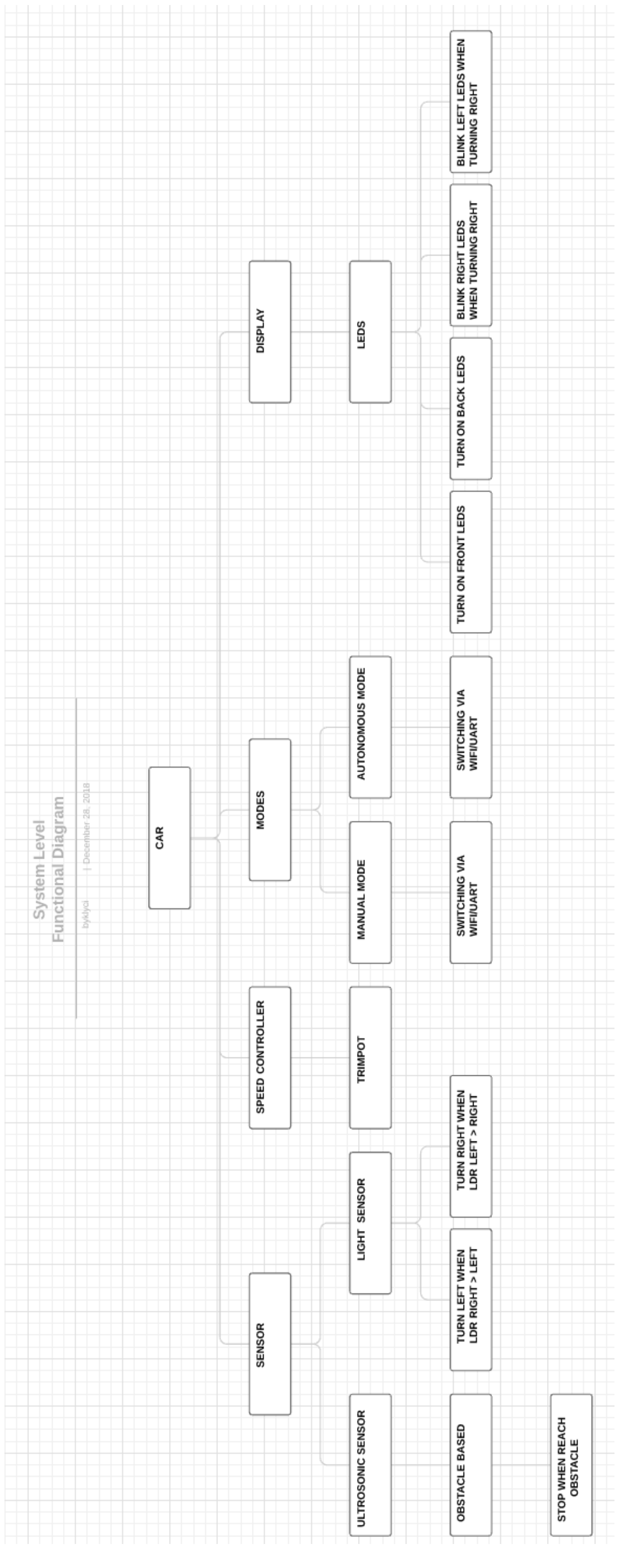

Figure 3: System-Level Functional Diagram

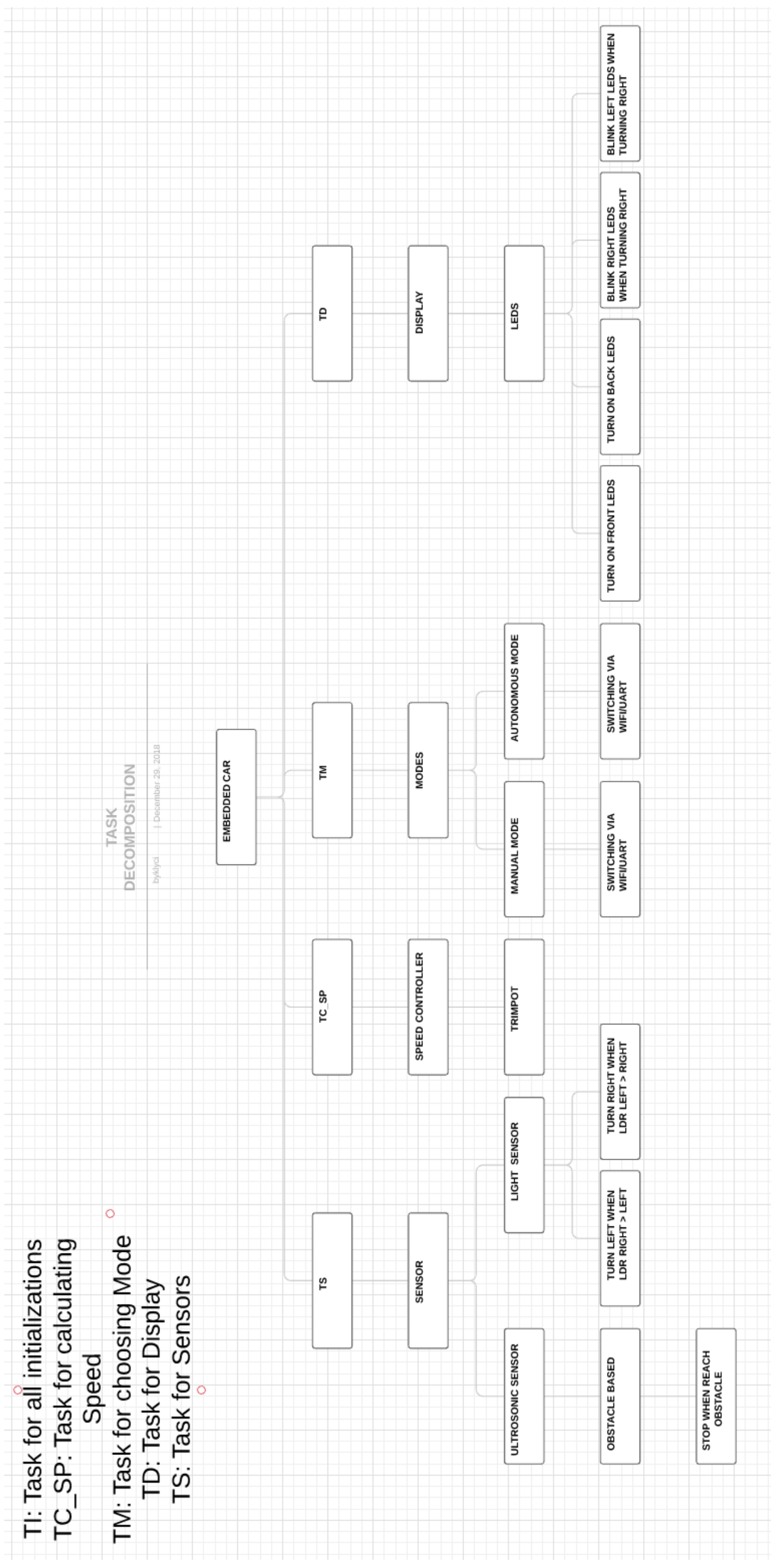

Figure 4: Task Decomposition

5 Sequence Diagrams

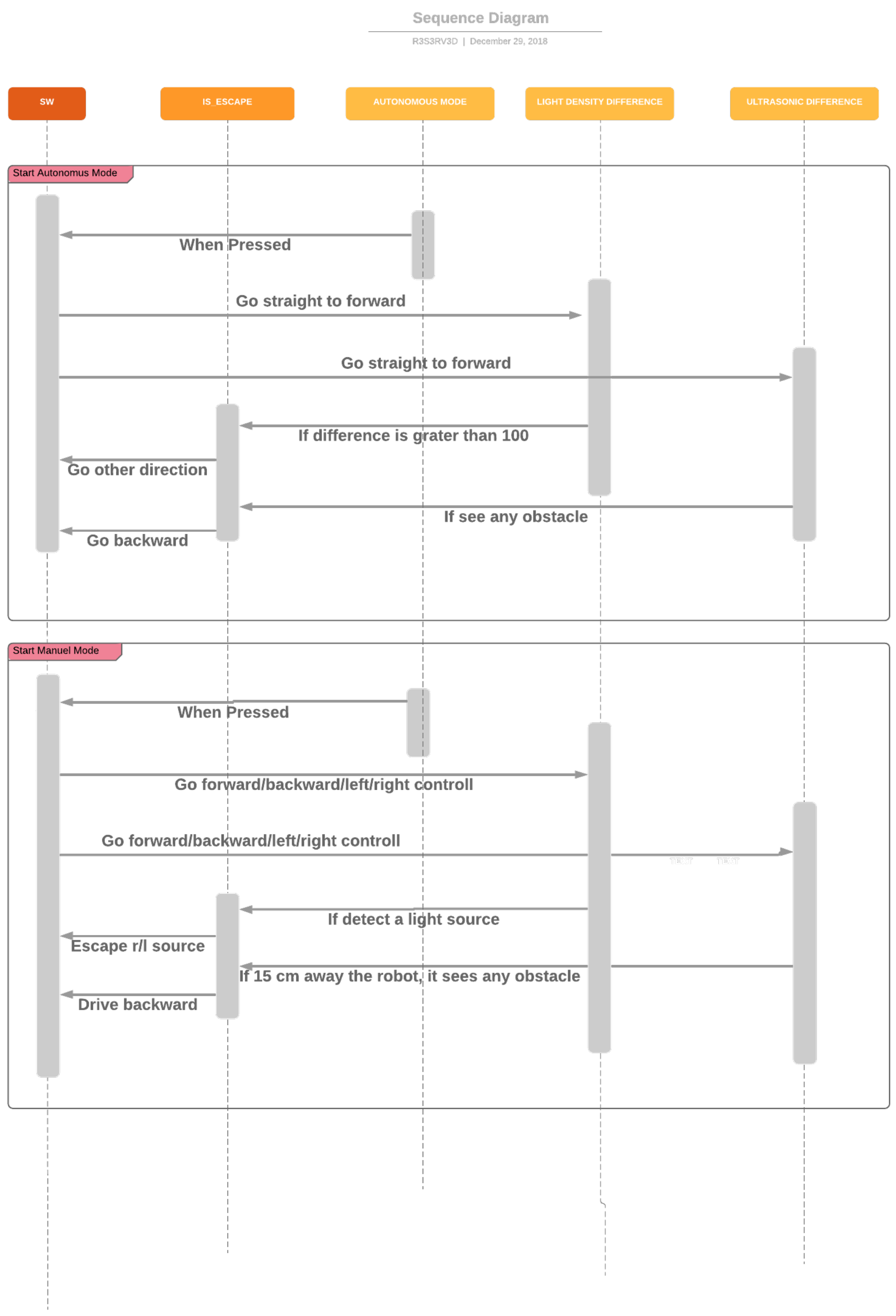

Figure 5: Sequence Diagram

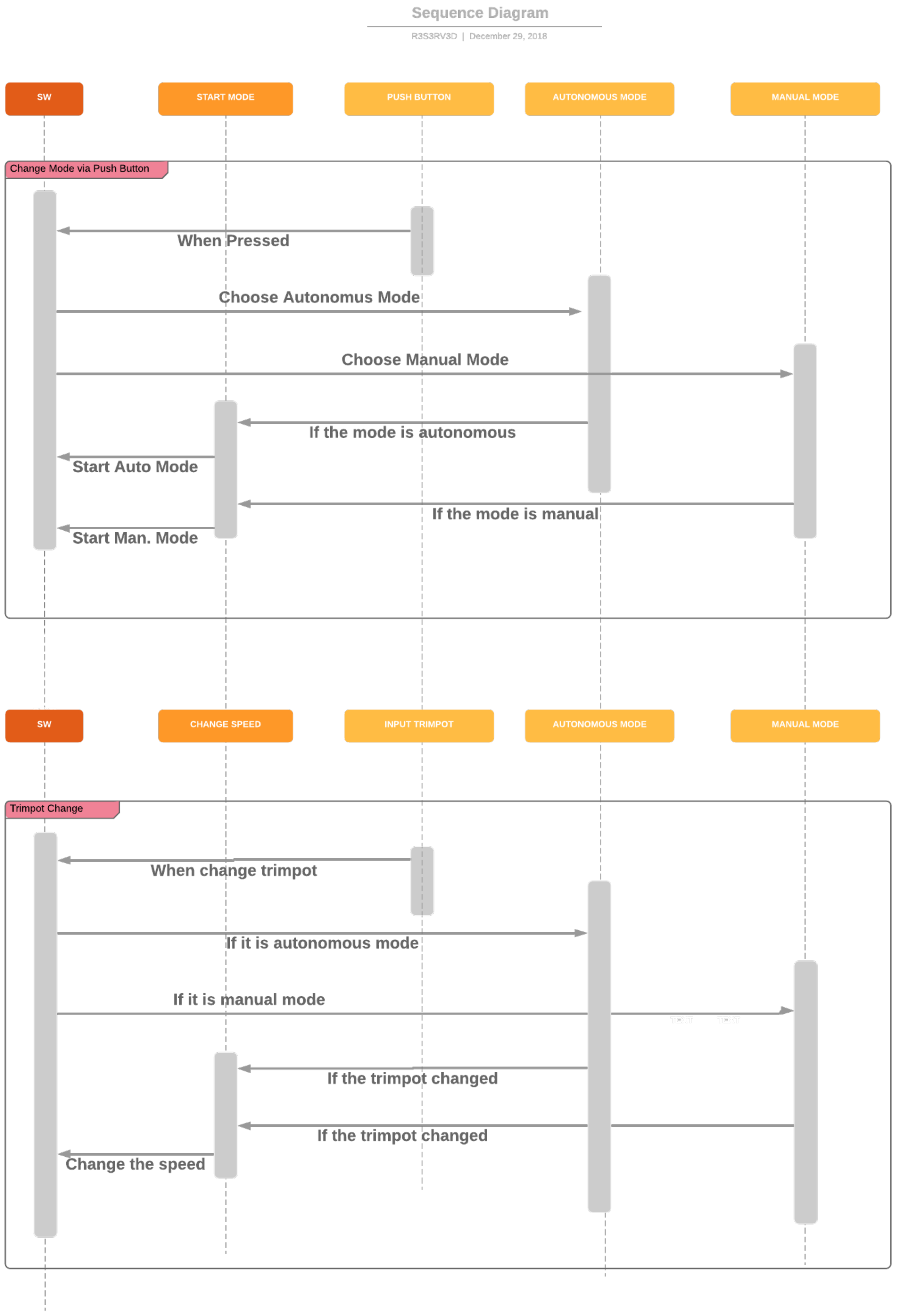

Figure 6: Sequence Diagram

6 Connections

6.1 LDR - Driver Connections

The LDR Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
Left LDR	P0.26 (P18)	ADC[3]	To get the Analog value from the LDR and read as Digital in the Driver
Right LDR	P0.25 (P17)	ADC[2]	To get the Analog value from the LDR and read as Digital in the Driver

Table 1: Selected LDR Pins as Analog to Digital Converter

Two LDR's are used one for the front right of the car and one for the front left. Analog to Digital converter pins are used to convert the resistor values which are analog and turn to digital for the computer to execute.

6.2 Ultrasonic - Driver Connections

Ultrasonic Pin Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
ECHO	P0.24 (P16)	T3_CAP_1	To capture signal with Timer 3
TRIGGER	P0.9 (P11)	T2_MAT_3	To manages sensor's signals. It is External Match Output.

Table 2: Selected Ultrasonic Pins

Ultrasonic Sensor is used to determine whether there is any obstacle in the given range. The distance is calculated from the capture and match values and action is taken accordingly.

6.3 WIFI - Driver Connections

WIFI Pin Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
CH_PD	P0.8 (P12)	GPIO	To enable WiFi module's communication
RST	P0.7 (P13)	GPIO	To reset WiFi module
RX	P0.0 (P9)	UART3_TX	To send data to WiFi module
TX	P0.1 (P10)	UART3_RX	To get data from WiFi module

Table 3: Selected WIFI Pins

6.4 Buttons & Controllers - Driver Connections

Button Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
Push Button	P23	EINT0	To generate External interrupt
Trimpot	P0.23 (P15)	ADC[0]	To change car speed between 0-100.
Joystick Up	P39	GPIO	It's already connected.
Joystick Down	P38	GPIO	It's already connected.
Joystick Right	P37	GPIO	It's already connected.
Joystick Left	P32	GPIO	It's already connected.
Joystick Center	P1	GPIO	It's already connected.

Table 4: Selected Button and Controller Pins (Already Installed on Board)

- Push Button which is already installed on the board controls the Mode of the Car with the related EINTO pin.
- Trimpot which is already installed on the board sets the speed of the car with the related Analog to Digital Converter.
- The Joystick which is already installed on the board controls the movement of the Car with it's 5 different pins initialized as GPIO's.

6.5 Motor - Driver Connection

Motor Terminal	**Motor Driver Terminal**
Motor FL +	Output A +
Motor FL -	Output A -
Motor BL +	Output A +
Motor BL -	Output A -
Motor FR +	Output B +
Motor FR -	Output B -
Motor BR +	Output B +
Motor BR -	Output B -

Table 5: Motor - Driver Connection

A 2-Channel Motor Controller is used to control 4 Motors. Since the right wheels and the left wheels needs to move simultaneously between theirselves, the BL & FL motors are connected to the same ports and the BR & FR motors are connected to the same ports on the Motor Controller.

6.6 Driver - Board Connection

Motor Driver Pin Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
ENA	P1.11 (P25)	PWM0[5]	Since we chose PWM1 for LEDs in order to prevent any conflicts, we chose PWM0 for motor.
ENB	P1.7 (P26)	PWM0[6]	Since we chose PWM1 for LEDs in order to prevent any conflicts, we chose PWM0 for motor.
IN1	P1.23 (P6)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. It is used to control channel A motors' direction.
IN2	P1.24 (P5)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. It is used to control channel A motors' direction.
IN3	P0.5 (P33)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. It is used to control channel A motors' direction.
IN4	P0.4 (P34)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. It is used to control channel A motors' direction.

Table 6: Driver - Board Connection

6.7 LED - Driver Connections

The LED Terminal	**LPC4088 Pin**	**Pin Functionality**	**Reason**
Front-left (-)	P1.6 (P27)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. Also, it controls ON/OFF state of LED.
Front-right (-)	P0.21 (P8)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. Also, it controls ON/OFF state of LED.
Back-left (-)	P1.30 (P19)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. Also, it controls ON/OFF state of LED.
Back-right (-)	P1.31 (P20)	GPIO	It is a GPIO pin which is not conflicting with any PWM or Timer functionalities. Also, it controls ON/OFF state of LED.
Front-left (+), Front-right (+), Back-left (+), Back-right (+)	P1.20 (P7)	PWM1[2]	It has PWM functionality. It provides BLINKING LEDs.

Table 7: Selected LED pins

Note: The + (positive) poles of the LED's are connected to GPIO's and the - (negative) poles are connected to a PWM connection on the breadboard instead of GND. We did not need 4 different PWM's for every single LED since they should blink as the same, instead we connected all the LED's in one PWM connection and controlled the ON & OFF situations from their related GPIO pins.
By this, we achieved a simpler design and the LED's blink harmoniously. You can see the circuit level design from Figure 7.

7 Circuit Schematics

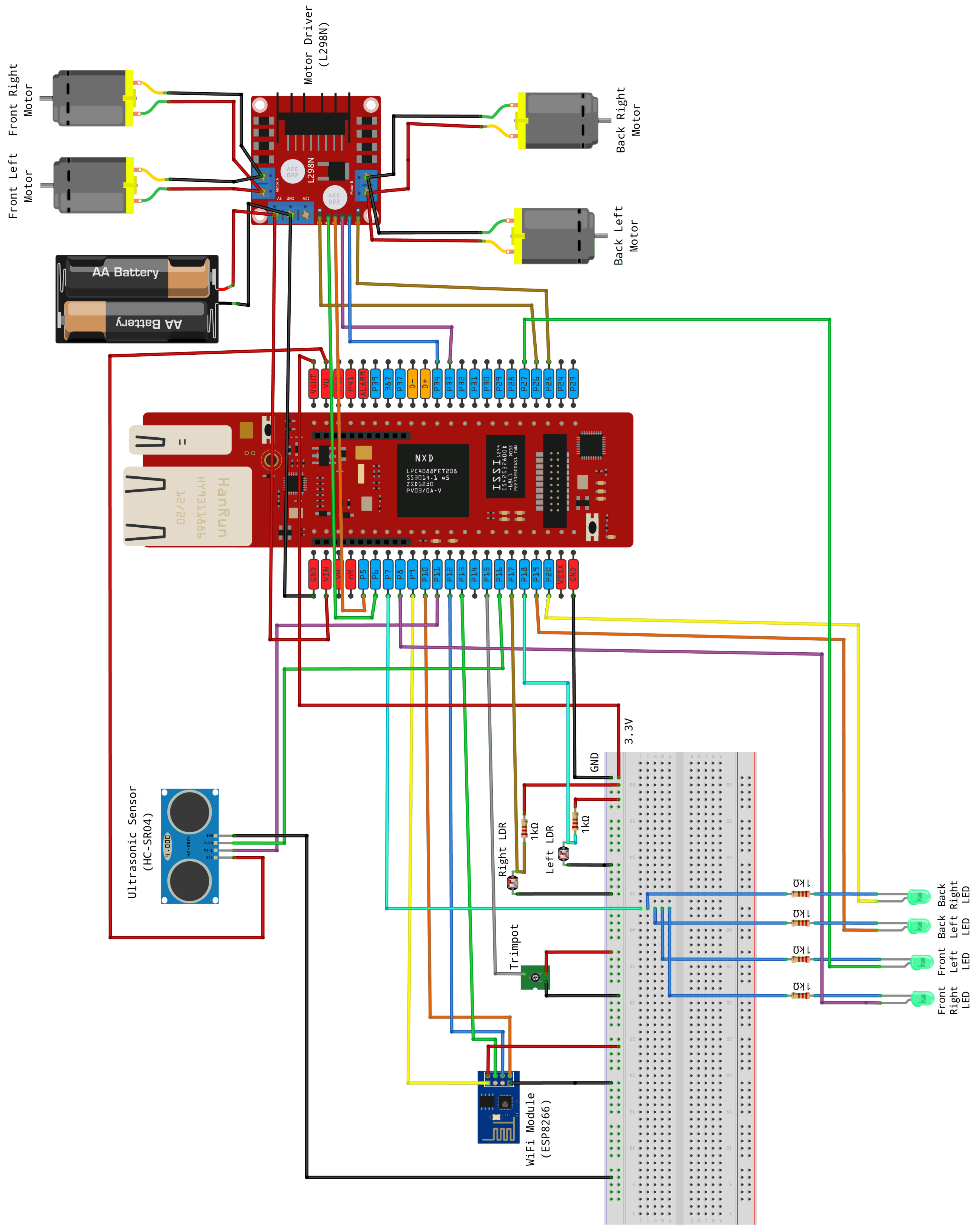

Figure 7: Circuit Diagram

8 Pseudocode

The Pseudocode represantation of our functions implemented (other than the init methods) is given in this section. Detailed explanation of the function is given in comment section on its top.

```
///////////////////////////////////////////
// Start Point tof the all system
function init:
 call Car_Init
///////////////////////////////////////////
// Initilization of the Functions
funtion Car_Init:
 call Joystick_Init
 call Motor_Init
 call LED_Init
 call Ultrasonic_Trigger_Timer_Init
 call Ultrasonic_Capture_Timer_Init
 call Ultrasonic_Start_Trigger
 call Trimpot_Init
 call LDR_Init
 call External_Init
 call Serial_Init
///////////////////////////////////////////
// Set the Controller of the LED and Motor and set flags for going
 ↪  forward
funtion goForward:
 call LED_Controller(ON, ON, OFF, OFF, 0)
 call Motor_Controller(FORWARD, FORWARD, speed)
 call setFlags(0, 0, 1, 0)
///////////////////////////////////////////
// Set the Controller of the LED and Motor and set flags for going
 ↪  backward
function goBackward:
 call LED_Controller(OFF, OFF, ON, ON, 0)
 call Motor_Controller(BACKWARD, BACKWARD, speed)
 call setFlags(0, 0, 0, 1)
///////////////////////////////////////////
// Set the Controller of the LED and Motor and set flags for going
 ↪  Right
function turnRight(rate ,flag):
 call LED_Controller(OFF, ON, ON, OFF, 1)
 call Motor_Controller(FORWARD, STOP, speed)
 call if (flag) then setFlags(0, 1, 0, 0)
///////////////////////////////////////////
```

```
// Set the Controller of the LED and Motor and set flags for going
↪  left
function turnLeft(rate ,flag):
 call LED_Controller(ON, OFF, OFF, ON, 1)
 call Motor_Controller(STOP, FORWARD, speed)
 call if (flag) then setFlags(1, 0, 0, 0)
///////////////////////////////////////////
// Set the Controller of the LED and Motor and set flags for going
↪  stoppng
function stopCar:
 call LED_Controller(OFF, OFF, OFF, OFF, 1)
 call Motor_Controller(STOP, STOP, 0)
 callsetFlags(0, 0, 0, 0)
///////////////////////////////////////////
// Get the values from the gadget and send them to the variables and
↪  set them
function updateSensorValues(void):
 set speed=Trimpot_Read_Data
 if(turnCount % 500 == 0) then {
  set distance = Ultrasonic_Get_Distance
  set turnCount = 0
 }
 set LDR_Left_Value  = LDR_Left_Read_Data
 set LDR_Right_Value = LDR_Right_Read_Data
 set LDR_Difference  = abs(LDR_Left_Value - LDR_Right_Value)

 set turnCount += 1;
///////////////////////////////////////////
// Change the car mode From autonomous too manual or from manual to
↪  autonomous
function toggleMode(void):
 if (mode == AUTO) then
 set mode = MANUEL;
 else if (mode == MANUEL) then
 set mode = AUTO;
 call stopCar();
 set active=0;

 call return mode;

///////////////////////////////////////////
// Set the car mod as a autonumous or manual
function setMode(nMode):
 call stopCar
 set active = 0
 set mode = nMode
//////////////////////////////////////////
// Set it is escaping from obstacle or not
function startEscape{
 set is_escaping=1
 call gobackward
//////////////////////////////////////////
// Stop the escaping mode
function endEscape:
 set IS_ESCAPING = 0
 call goForward
//////////////////////////////////////////
// Check for obstacle:if it is close start escaping
if it is enoughly far from the obstacle stop escaping
function checkObstacle(void):
 if (!IS_ESCAPING && isMoving() && distance < OBSTACLE_DISTANCE) then
  call startEscape
  else if (IS_ESCAPING && distance > OBSTACLE_ESCAPE_DISTANCE &&
   ↪  distance < ULTRASONIC_MAX_DISTANCE) then
  call endEscape
//////////////////////////////////////////
// function setFlags(turnLeft, turnRight, forward, backward):
Set the flag of the directions
 set TURN_LEFT_FLAG  = turnLeft
 set TURN_RIGHT_FLAG = turnRight
 set FORWARD_FLAG    = forward
 set BACKWARD_FLAG   = backward
//////////////////////////////////////////
// return 1 if any direction flag is 1
function isMoving:
 call return((TURN_LEFT_FLAG + TURN_RIGHT_FLAG + FORWARD_FLAG +
  ↪  BACKWARD_FLAG) > 0;)
//////////////////////////////////////////
// Control the car on the autonomous mode.There is some flags and some
 ↪  controls for holding the car on the road.
function update_auto:
 if (Joystick_Center_Pressed) then :
   set active = 0
   call stopCar
 else if (Joystick_Up_Pressed) then:
   set active = 1;
 if (active) then :
  if (LDR_Difference > DIFFERENCE_THRESHOLD) then :
   set speed = TURN_SPEED
   if (LDR_Left_Value > LDR_Right_Value) then :
    call turnLeft(LDR_Difference, 1);
   else :
    call turnRight(LDR_Difference, 1);
  else :
   set goForward
//////////////////////////////////////////
// This method is written for manual mode.There is some flags and some
 ↪  controls for holding the car on the road.
function update_manual
if(isMoving then :
  if ( not IS_ESCAPING && distance < OBSTACLE_DISTANCE) then :
   call startEscape
     else if (IS_ESCAPING && distance > OBSTACLE_ESCAPE_DISTANCE &&
      ↪  distance < ULTRASONIC_MAX_DISTANCE) then :
   call endEscape
  if (LDR_Left_Value < LIGHT_THRESHOLD) then :
   call turnRight
  else if (LDR_Right_Value < LIGHT_THRESHOLD) then :
   turnLeft
  else if (FORWARD_FLAG) then :
   goForward();
  else if (BACKWARD_FLAG) then :
   goBackward();

 if Joystick_Center_Pressed then :
  call stopCar
 else if Joystick_Up_Pressed then :
  call goForward
 else if Joystick_Down_Pressed then :
  call goBackward
 else if Joystick_Left_Pressed then :
  call turnLeft
 else if Joystick_Right_Pressed then :
  call turnRight

//////////////////////////////////////////
// This method is written for selecting the mode of the car.
function update:
 if (mode == AUTO) then :
  call update_auto
 else if (mode == MANUEL) then :
  call update_manual

//////////////////////////////////////////
UART_IRQHandler:
 set currentInterrupt = ((Serial_UART->IIR & (0x7 << 1)) >> 1);

 if (currentInterrupt == 0x02) then :
  set serialReceivedCharacter = Serial_ReadData
  if(serialReceivedCharacter == AUTO) then :
   call Serial_Write("AUTO\r\n");
   call setMode(AUTO);
  else if(serialReceivedCharacter == MANUEL) then :
   call Serial_Write("MANUEL\r\n");
   call setMode(MANUEL);
  else if (serialReceivedCharacter == '6') then :
   if(serialLastCharacter == '6') then :
    set serialLastCharacter = 0
    if(mode == AUTO) then :
     call Serial_Write("STARTED\r\n")
     set active = 1
    else :
     call Serial_Write("NOT STARTED\r\n")
   else :
    set serialLastCharacter = '6';
 else if (currentInterrupt == 0x1) then :
  //Second if statement is for THRE interrupt
  if (*serialTransmitData > 0) then :
   call Serial_WriteData(*serialTransmitData++)
  else :
   set serialTransmitCompleted = 1

///////////////////////////////////////////
// Call the initilization code of the MotorPWM and MotorDirection
function Motor_Init:
 call Motor_PWM_Init
 call Motor_Direction_Init

///////////////////////////////////////////
// Init the MotorPWM and MotorDirection of the car.
function Motor_PWM_Init:
 call PWM_Init(PWM0, &IOCON_LEFT_MOTOR, IOCON_LEFT_MOTOR_PWM_FUNC,
  ↪  PWM0_PCONP, LEFT_MOTOR_PWM_CHANNEL);
 call PWM_Init(PWM0, &IOCON_RIGHT_MOTOR, IOCON_RIGHT_MOTOR_PWM_FUNC,
  ↪  PWM0_PCONP, RIGHT_MOTOR_PWM_CHANNEL);

///////////////////////////////////////////
// Initilization method for direction of the motor and motor
 ↪  controller
function Motor_Direction_Init:
 call GPIO_DIR_Write(IN1_PORT, IN1_MASK, OUTPUT)
 call GPIO_DIR_Write(IN2_PORT, IN2_MASK, OUTPUT)
 call GPIO_DIR_Write(IN3_PORT, IN3_MASK, OUTPUT)
 call GPIO_DIR_Write(IN4_PORT, IN4_MASK, OUTPUT)
 call Motor_Controller(STOP, STOP, 0)
///////////////////////////////////////////
// Function is used for determining the channel pwm and speed
function Motor_Write:
 call PWM_Write(PWM0, channel, abs(speed))
///////////////////////////////////////////
// Change the Motor controller name according to the percentage of the
↪  LDR.
function Motor_Handle(percentageOfLDR, PIN of 1,PIN of 2):
 if (percentageOfLDR > 0) then:
  call GPIO_PIN_Write(MOTOR1, MASK1, LOW)
  call PIO_PIN_Write(MOTOR2, MASK2, HIGH)
 else if (percentage < 0) then:
  call GPIO_PIN_Write(MOTOR1, MASK1, HIGH)
  call GPIO_PIN_Write(MOTOR2, MASK2, LOW)
 else if (percentage == 0) then:
  call GPIO_PIN_Write(MOTOR1, MASK1, LOW)
  call GPIO_PIN_Write(MOTOR2, MASK2, LOW)
///////////////////////////////////////////
// Change the rotation value of the car according to the LDR values.
function Motor_Controller(leftPercentageOfLDR, rightPercentageOfLDR,
↪  speedOfCar):
 // Set direction of motors
 call Motor_Handle(leftPercentageOfLDR,setEnableA,setEnableB)
 call Motor_Handle(rightPercentageOfLDR,setEnableC,setEnableD)

 // Set speed of motor
 call Motor_Write(leftPercentageOfLDR * speed / 100,
  ↪  LEFT_MOTOR_PWM_CHANNEL)
 call Motor_Write(rightPercentageOfLDR * speed / 100,
  ↪  RIGHT_MOTOR_PWM_CHANNEL)
///////////////////////////////////////////
// Set the led on or off accroding to the flags of the directions. If
↪  all LEDs are OFF, set duty cycle to 0. If they are blink, change
↪  cycle rate for PMW1

function LED_Controller(frontLeft, frontRight, backRight, backLeft,
↪  isBlink):
 if (frontLeft) then :
  LED_On(front left led)
 else :
   LED_Off(front left led)
 if (frontRight) LED_On(front right led) then :
 else :
LED_Off(front right led)
 if (backRight) LED_On(back right led) then :
 else :
  LED_Off(back right led)
 if (backLeft) LED_On(back left led) then
      else LED_Off(back left led)

 // If all LEDs are OFF, set duty cycle to 0
 if (frontLeftLed is Lighting + frontRightLed is Lighting +
  ↪  backRightLed is Lighting + backLeftLed is Lighting == 0)then:
   call LED_Write(0)
 else :
  LED_Write(60)
 // If they are blink, change cycle rate for PMW1
 if (leds are blinking) then :
   call PWM_Cycle_Rate(PWM1, 500)
 else:
   call PWM_Cycle_Rate(PWM1, 20)

///////////////////////////////////////////
```

Code 1: Pseudo Code

9 Conclusion

9.1 Expense List

No expense is done for the project since we found cables from our friends for free and no other modifications were made. All of our equipments were from the Hardware LAB of CmpE Boğaziçi University.

9.2 Outcomes

The R3S3RV3D Car is working properly. All the basic functions and requirements on the Final Project are complete.
With this project we achieved skills like:

- Determining the correct pins on a Microcontroller Board with the desired functionality.
- Contolling multiple LEDs.
- Controlling and Installing multiple DC motors.
- Getting feedbacks for DC motors and acting accordingly.
- Using the Ultrasonic Sensor to detect an obsatcle.
- Using Light Sensors (LDR's) to read the light density.
- Getting Analog signals and convert to Digital.
- Using a Timer in a complicated system.
- Using the PWM channel in different needs. (Motor & LED's)
- Using UART functions.
- Using WIFI (even if not every task is complete we do connect and communicate)
- Working as a team.
- Designing the BreadBoard.
- Learning that jumper cables are hard to find and the best way to get one is to go to Karaköy :)