CmpE 443 Final Project Design Document
Group Name : Doğan SLX

Members
Emirhan SARAÇ (Team Leader)
Muhammed Fatih BALIN
Yusuf BAŞPINAR
Enis SİMSAR

December 28, 2018
Version 1.00

Contents

1 Introduction

Our GitHub Resository:
https://github.com/443-Embedded/embedded-final-project
Doğan SLX component is an embedded system for a car with joystick controller for manual mode or autonomous mode. Joystick, UART and WIFI is an input type component. It takes the input from user and transfer it to the board. We used two motor speed sensor for each side: right and left. We have LEDs, LDRs and Motor Driver Controller components for output. Actions:

1.1 Manual Mode

- The robot will be at stop phase (output), when robot enters the manual mode (input, push button or UART or WIFI).
- When Joystick Left button is pressed (input), starts in counter-clockwise direction (output) and blink left LEDs (output).
- When Joystick Up button is pressed (input), starts to travel in forward direction (output) and turn on front LEDs (output).
- When Joystick Down button is pressed (input), starts to travel in backward direction (output) and turn on back LEDs (output).
- When Joystick Center button is pressed (input), stops (output).
- When Joystick Right button is pressed (input), starts in clockwise direction (output) and blink right LEDs (output).
- While going in forward direction, if ultrasonic sensor detects an obstacle (its direction is parametric in parameters.h file) (output for trigger pin and echo for echo pin), robot moves backward (output) until a length which is parametric and stop (output).
- Trimpot resistor can change the speed of the robot. (getting information input, adjusting the speed according to its value output)/
- If the robot detect a light source while going in forward direction (input from LDRs), it escape from the light source (adjusting leds and motor speeds and directions.) (output)

1.2 Auto Mode

- The robot will be at stop phase (output), when robot enters the auto mode (input, push button or UART or WIFI).
- The robot will be start with two different ways: pressing joystick up button (input) or sending 66 via UART or WIFI (input). Then, robot will be start.

(output) (forward leds and forward direction movement). The robot go along the road.

- While going in forward direction, if ultrasonic sensor detects an obstacle (its direction is parametric in parameters.h file) (output for trigger pin and echo for echo pin), robot moves backward (output) until a length which is parametric and stop (output).
- Trimpot resistor can change the speed of the robot. (getting information input, adjusting the speed according to its value output)/
- If the robot detect a light source while going in forward direction (input from LDRs), it escape from the light source (adjusting leds and motor speeds and directions.) (output)

2 Block Diagram

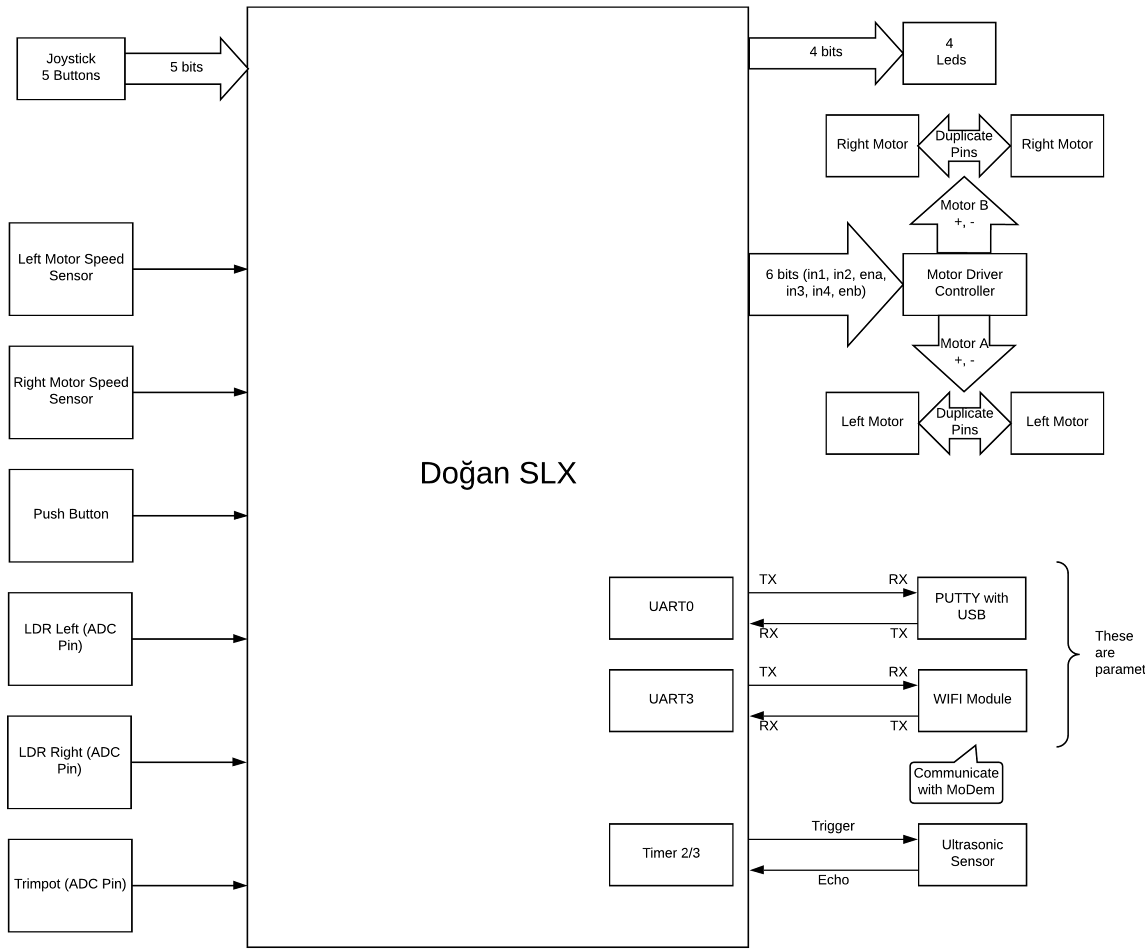

Figure 1: Block Diagram

3 System-Level Functional Diagram

For big version click here.

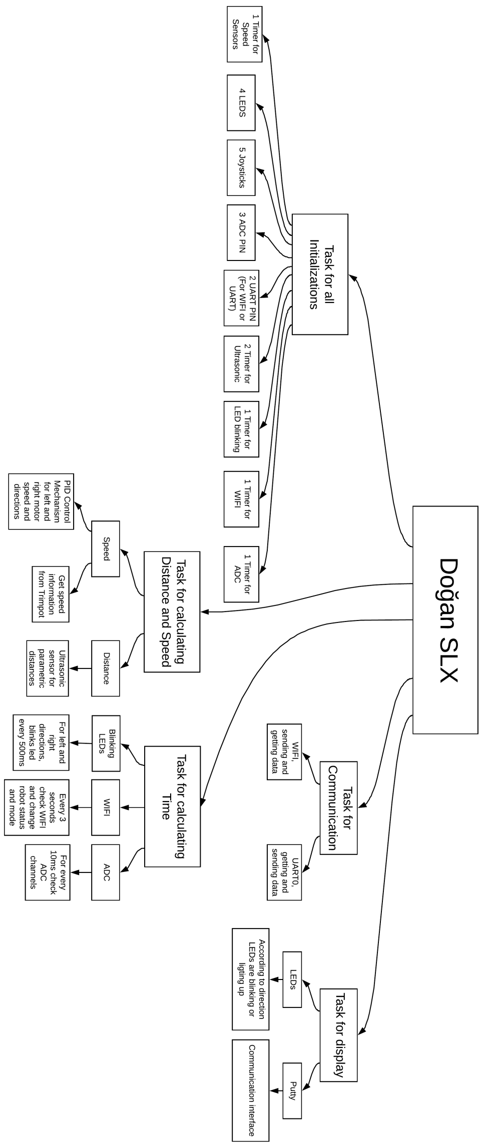

Figure 2: System-Level Functional Diagram

4 Sequence Diagram

4.1 Manual Mode Diagrams

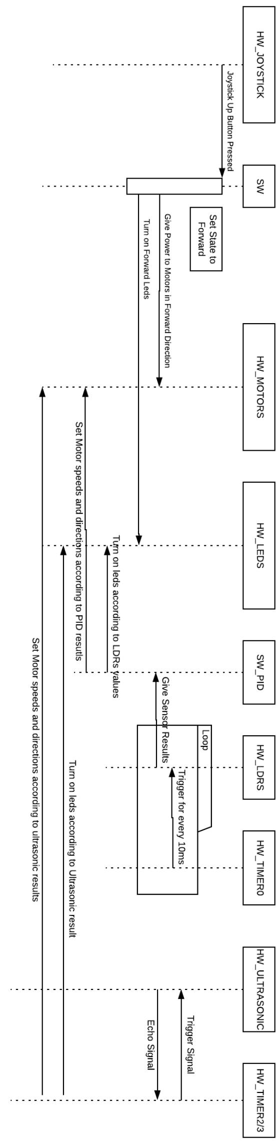

Figure 3: Forward Sequence Diagram

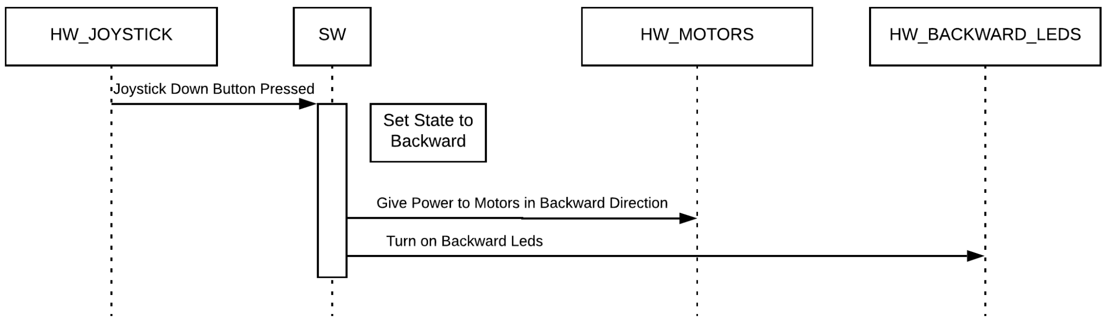

Figure 4: Backward Sequence Diagram

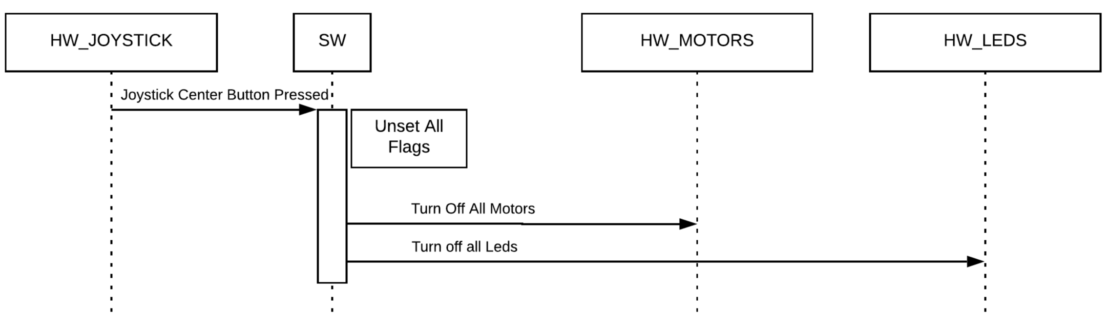

Figure 5: Stop Sequence Diagram

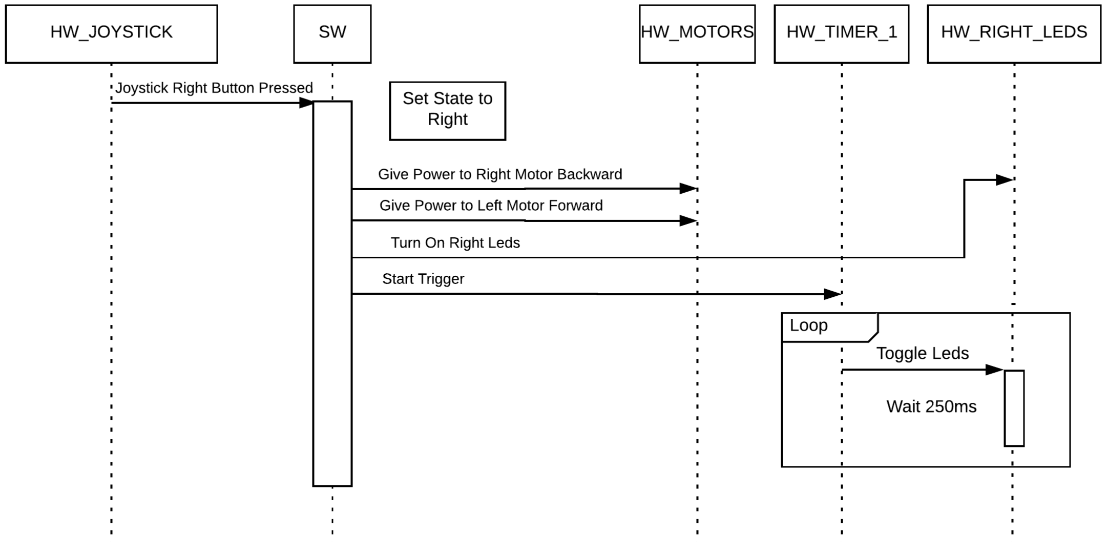

Figure 6: Right Sequence Diagram

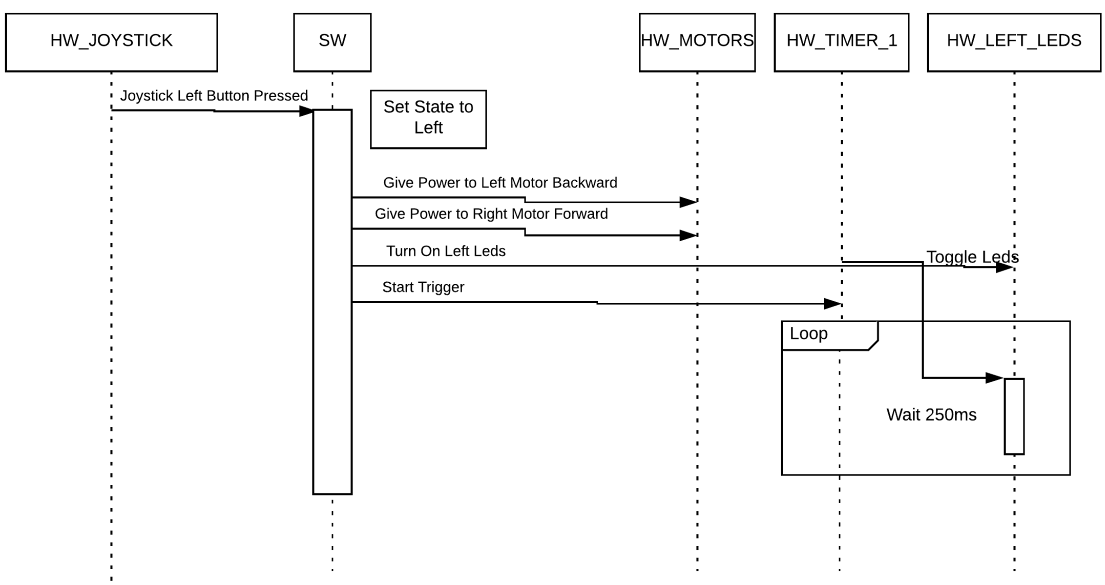

Figure 7: Left Sequence Diagram

4.2 Auto Mode Diagrams

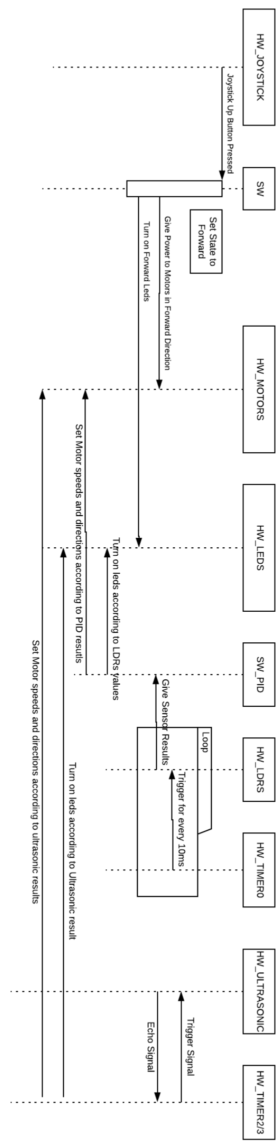

Figure 8: Forward Sequence Diagram for AUTO Mode

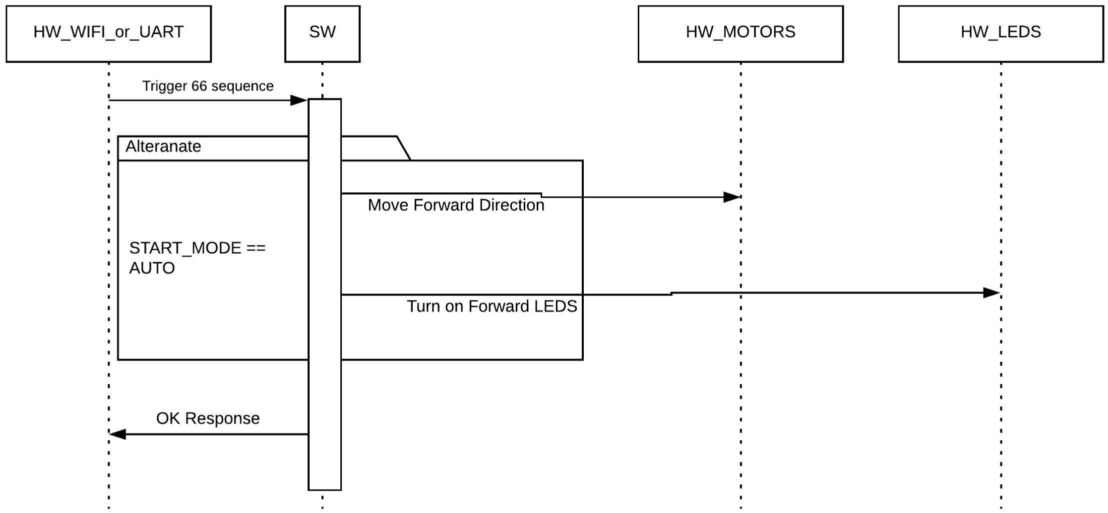

Figure 9: Start 66 Signal Sequence Diagram

4.3 Both Mode Diagrams

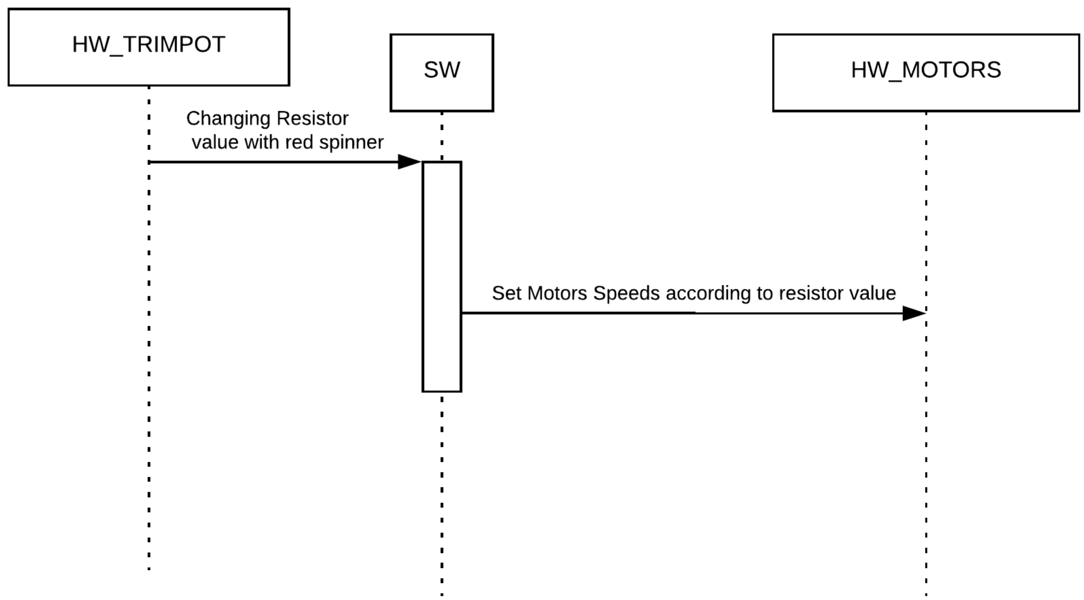

Figure 10: Trimpot Sequence Diagram

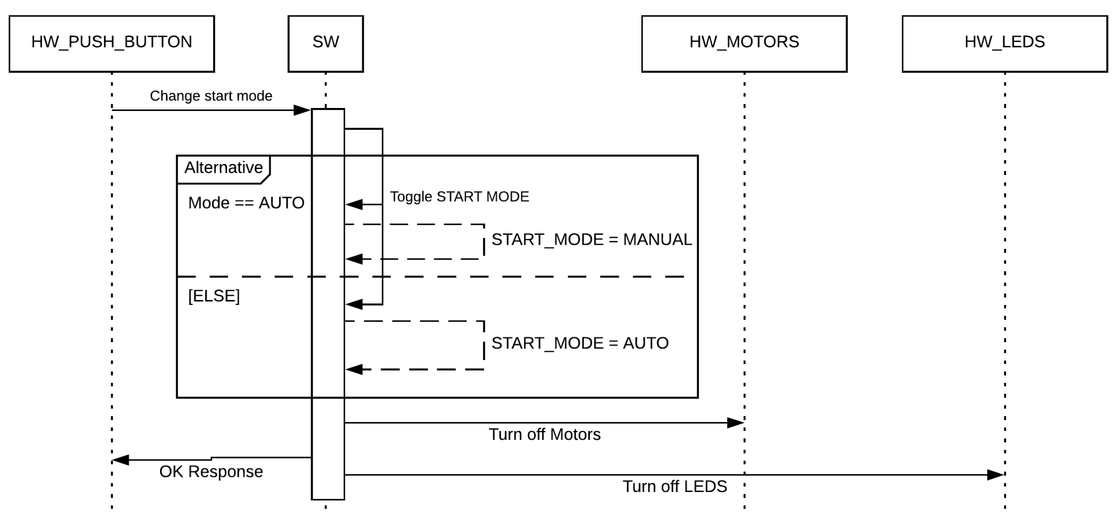

Figure 11: Push Button Sequence Diagram

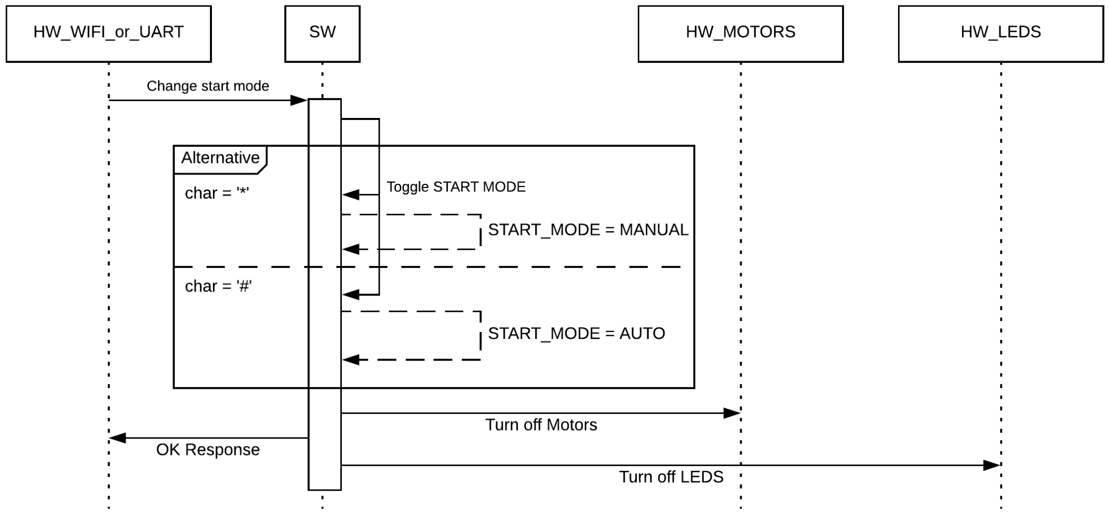

Figure 12: WIFI or UART Sequence Diagram

5 LED Connections

LED Name	LPC4088 Pin	Pin Functionality	Reason
Forward Left	P1.5	GPIO	Close 4 Pins
Forward Right	P1.6	GPIO	Close 4 Pins
Backward Right	P1.7	GPIO	Close 4 Pins
Backward Left	P1.11	GPIO	Close 4 Pins

Table 1: LED Connections

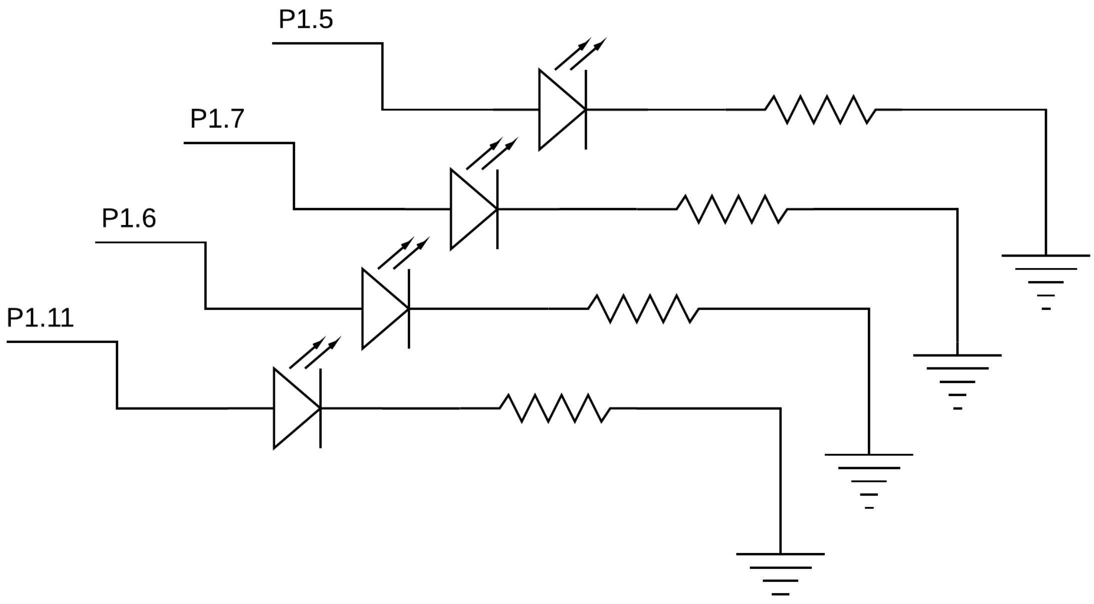

Figure 13: LED Circuits

6 Motor - Speed Sensor Connection

Speed Sensor Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
Speed Sensor Left	P0.4	T2_CAP0	Availability of only input for Timer 2, channel 0.
Speed Sensor Rigth	P0.5	T2_CAP1	Availability of only input for Timer 2, channel 1.

Table 2: Sensor Table

7 Motor - Driver Connection

Motor Terminal	**Motor Driver Terminal**
Motor A Red (Left Motors +)	Output A+(OUT1)
Motor A Black (Left Motors -)	Output A-(OUT2)
Motor B Red (Right Motors +)	Output B+(OUT3)
Motor B Black (Right Motors -)	Output B-(OUT4)

Table 3: Motor - Driver Table

8 Driver - Board Connection

Motor Driver Pin Name	**LPC4088 Pin**	**Pin Functionality**	**Reason**
EnA	P1.2	PWM0_1	Adjacent 2 PWM0 Pins
In1	P0.9	GPIO	Adjacent 4 GPIO Pins
In2	P0.8	GPIO	Adjacent 4 GPIO Pins
EnB	P1.3	PWM0_2	Adjacent 2 PWM0 Pins
In3	P0.1	GPIO	Adjacent 4 GPIO Pins
In4	P0.0	GPIO	Adjacent 4 GPIO Pins

Table 4: Motor Driver Connections

9 Ultrasonic Sensor - Board Connection

Ultrasonic Sensor Pin Name	**LPC4088 Pin**	**Pin Functionality**
Trigger	P0.7	T2_MAT1
Echo	P0.24	T3_CAP1

Table 5: Ultrasonic Sensor Connections

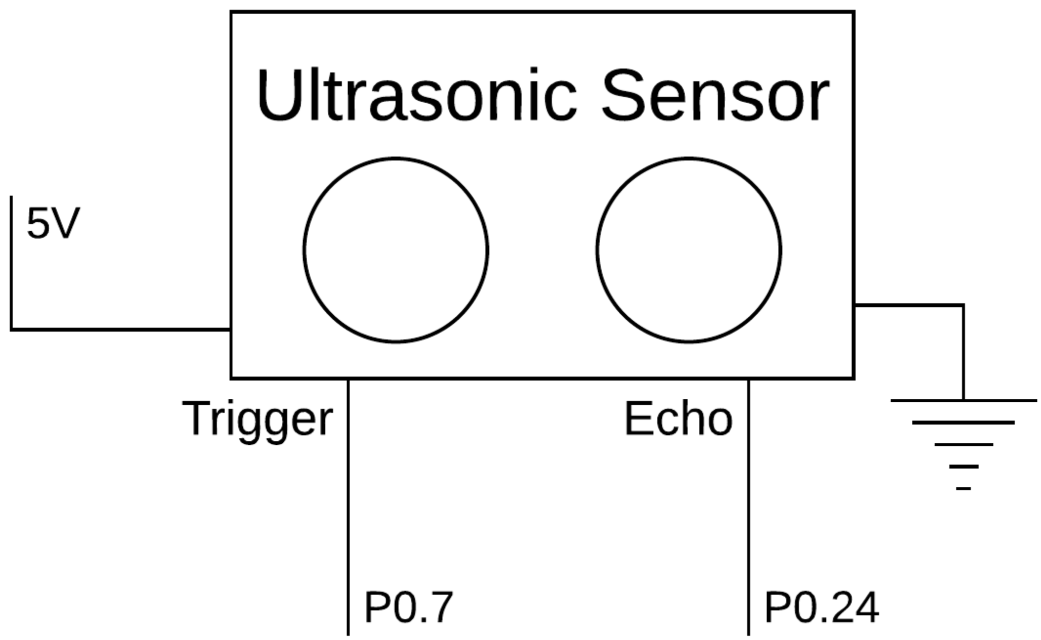

Figure 14: Ultrasonic Sensor Circuits

10 LDR Resistors - Board Connection

LPC4088 Pin	Pin Functionality	Name
P0.25	ADC Channel 2	Left LDR
P0.26	ADC Channel 3	Right LDR

Table 6: LDR Resistors Connections

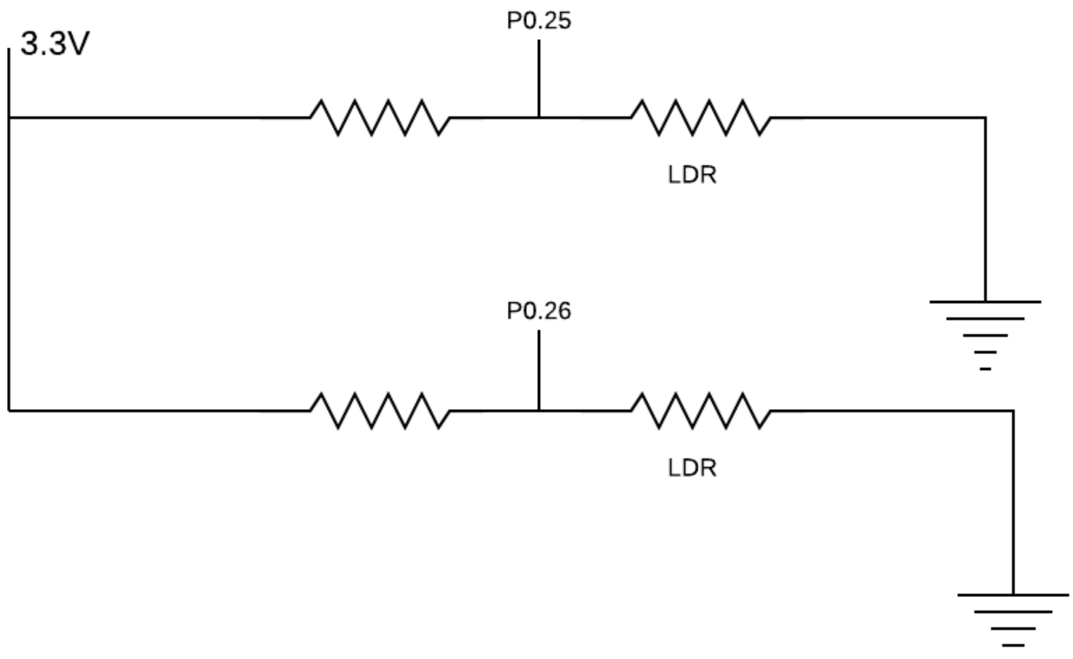

Figure 15: LDR Circuits

11 Push Button & Trimpot Resistors - Board Connection

LPC4088 Pin	**Pin Functionality**	**Name**
P2.10	EINT_0	Push Button
P0.22	ADC Channel 0	Trimpot

Table 7: Push button and Trimpot Connections

12 WIFI/ UART - Board Connection

WIFI or UART Pin Name	**LPC4088 Pin**	**Pin Functionality**
TX	P0.3	U0_RX for UART, U3_RX for WIFI
RX	P0.2	U0_TX for UART, U3_TX for WIFI

Table 8: WIFI/ UART Connections

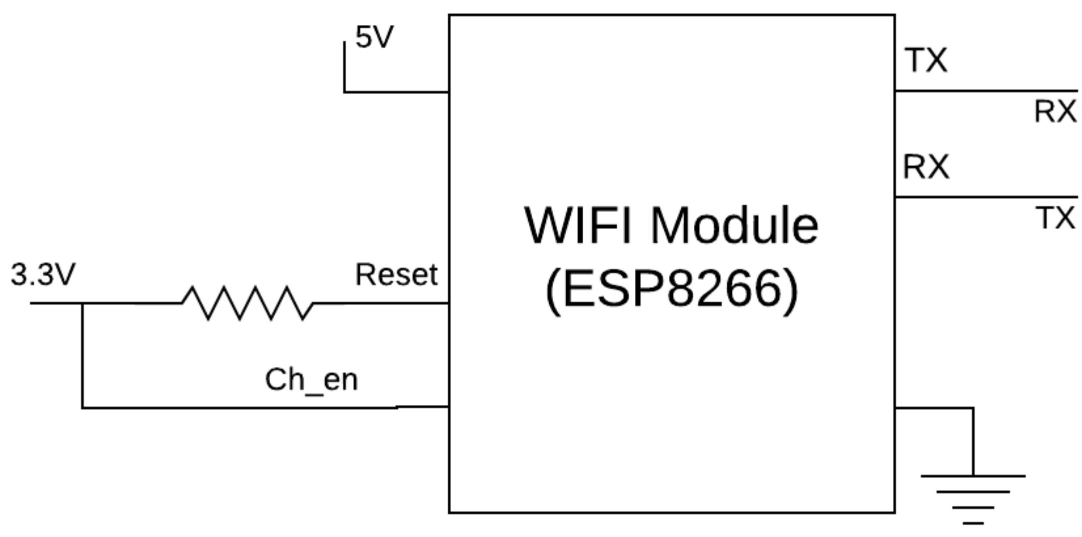

Figure 16: WIFI Circuits

13 Pin Connections - All

LPC4088 mbed	**LPC4088 Pin Name**	**Pin Functionality**	**Component**
P9	P0.0	GPIO	Motor Driver
P10	P0.1	GPIO	Motor Driver
P11	P0.9	GPIO	Motor Driver
P12	P0.8	GPIO	Motor Driver
P13	P0.9	T2_MAT1	Ultrasonic Sensor
P15	P0.8	ADC0_IN[0]	Trimpot
P16	P0.8	T3_CAP1	Ultrasonic Sensor
P17	P0.8	ADC0_IN[2]	Right LDR
P18	P0.8	ADC0_IN[3]	Left LDR
P42	P0.2	U0_TDX	UART Communication
P41	P0.3	U0_RDX	UART Communication
P42	P0.2	U3_TDX	WIFI Communication
P41	P0.3	U3_RDX	WIFI Communication
P34	P0.4	T2_CAP_0	Right Speed Sensor
P33	P0.5	T2_CAP_1	Left Speed Sensor
P30	P1.2	PWM0_1	EnA
P29	P1.2	PWM0_2	EnB
P28	P1.5	GPIO	LED
P27	P1.6	GPIO	LED
P26	P1.7	GPIO	LED
P25	P1.11	GPIO	LED

Table 9: Motor Driver Connections

14 Expense List

There is no extra component.

15 Conclusion

We implemented WIFI, UART and Basic(push button) communications. WIFI and UART use same pins so they do not work together. For determining speed and motor directions, we implemented a PID control mechanism. We had not enough time to optimize our hyper parameters (Kp, Ki, Kd). If we had adjusted these parameters, the robot would have work more accurately.

16 Pseudocodes

Algorithm 1 MAIN

```
procedure MAIN                                              ▷
    Init()
    Wait
    while true do
        if START_MODE == AUTO then
            if Joystick_Up_Pressed() then
                TURN_LEFT_FLAG ← 0
                TURN_RIGHT_FLAG ← 0
                BACKWARD_FLAG ← 0
                FORWARD_FLAG ← 1
                MOTORDirection(0, FORWARD);
                MOTORDirection(1, FORWARD);
                LEDAdjuster(FORWARD_LED);
            break;
    if START_MODE == MANUAL then
        Update()
```

Algorithm 2 Init

```
procedure INITIALIZE                                ▷
    GPIO_Init()
    PWM_Init()
    External_Init()
    if WIFI_COMM then
        ESP8266_Init()
        ConnectWIFI
    else
        if UART_COMM then
            Serial_Init()
    TIMER0_Init()
    TIMER1_Init()
    TIMER2_Init()
    TIMER3_Init()
    ADC_Init()
```

Algorithm 3 UPDATE

```
procedure UPDATE ▷
    if Joystick-Center-Pressed then
        Motor_Direction(1, STOP)
        Motor_Direction(0, STOP)
        PWM_MOTOR_Write(0, 0)
        PWM_MOTOR_Write(0, 1)
        Led_Adjuster(STOP_LED)
    if Joystick-Up-Pressed then
        Motor_Direction(1, FORWARD)
        Motor_Direction(0, FORWARD)
        PWM_MOTOR_Write(0, 0)
        PWM_MOTOR_Write(0, 1)
        Led_Adjuster(FORWARD_LED)
    if Joystick-Down-Pressed then
        Motor_Direction(1, BACKWARD)
        Motor_Direction(0, BACKWARD)
        Led_Adjuster(BACKWARD_LED)
    if Joystick-Right-Pressed then
        Motor_Direction(1, BACKWARD)
        Motor_Direction(0, FORWARD)
        LedAdjuster(RIGHT_BLINKER)
    if Joystick-Left-Pressed then
        Motor_Direction(1, FORWARD)
        Motor_Direction(0, BACKWARD)
        PWM_MOTOR_Write(0, 0)
        PWM_MOTOR_Write(0, 1)
        Led_Adjuster(LEFT_BLINKER)
```

Algorithm 4 Set Speed

```
procedure setSpeed                                          ▷
    if FORWARD_FLAG then
        scale(LEFT_LDR)
        scale(RIGHT_LDR)
        inc = pid(ADC_LEFT_LDR − ADC_RIGHT_LDR − 200)
        rightSpeed = ROBOT_SPEED + inc
        leftSpeed ← ROBOT_SPEED − inc
        if leftSpeed > 100 then
            rightSpeed ← rightSpeed ∗ 100/leftSpeed
            leftSpeed ← 100
            if rightSpeed < -100 then
                leftSpeed ← leftSpeed ∗ 100/ − rightSpeed
                rightSpeed ← −100
        if rightSpeed > 100 then
            leftSpeed ← leftSpeed ∗ 100/rightSpeed
            rightSpeed ← 100
            if leftSpeed < -100 then
                rightSpeed ← rightSpeed ∗ 100/ − leftSpeed
                leftSpeed ← −100
        setMotor(1, rightSpeed)
        setMotor(1, leftSpeed)
    else
        PWM_MOTOR_Write(ROBOT_SPEED, 0)
        PWM_MOTOR_Write(ROBOT_SPEED, 1)
```

Algorithm 5 ADC_IRQHandler

```
procedure Interrupt                                         ▷
    if ADC-DR0 then
        TRIMPOT ← ADC − DR[0] >> 4
    if ADC-DR2 then
        RIGHT_LDR ← ADC − DR[2] >> 4
    if ADC-DR3 then
        LEFT_LDR ← ADC − DR[3] >> 4
```

Algorithm 6 Set Motor

procedure SETMOTOR($MOTOR\_TYPE, speed$) ▷
  if speed $<$ 0 **then**
    MOTOR_Direction(MOTOR_TYPE, BACKWARD)
    $speed \leftarrow -speed$
  else
    MOTOR_Direction(MOTOR TYPE, FORWARD)
  if speed $>$ 100 **then**
    $speed \leftarrow 100$
  PWM_MOTOR_Write(speed, MOTOR TYPE)

Algorithm 7 PWM_MOTOR_WRITE

procedure MOTOR ▷
  if T_ON $>$ 100 **then**
    $T\_ON \leftarrow 100$
  if T_ON $==$ PWM0-MR0 **then**
    $T - ON++$
  if MOTOR_TYPE $==$ 0 **then**
    $PWM0 - MR1 \leftarrow T\_ON$
  if MOTOR_TYPE $==$ 0 **then**
    $PWM0 - MR2 \leftarrow T\_ON$
  $PWM0 - LER \leftarrow MOTOR\_TYPE + 1$

Algorithm 8 TIMER0_IRQHandler

procedure INTERRUPT ▷
  if COMM_TYPE $==$ WIFI_COMM and count $==$ 300 **then**
    Call procedure WIFI check
  ADC_START

Algorithm 9 TIMER1_IRQHandler

procedure INTERRUPT ▷
  if TURN_LEFT_FLAG $!=$ 0 **then**
    $TURN\_LEFT\_FLAG++$
    LED_Change (TURN_RIGHT_FLAG)
  Clear the interrupt flag for MAT channel 0 event

Algorithm 10 TIMER2_IRQHandler

```
procedure INTERRUPT                                              ▷
    if TIMER2-IR 1 then
        if TIMER2-MR1 == 10 then
            Change MR1 Register Value for 60000
            ultrasonicSensorEdgeCount ← 0
        else
            Change MR1 Register Value for 10
        Reset Timer Counter and Prescale Counter for Timer2
        Start timer again
        Remove reset on Timer2
        Clear IR Register Flag for Corresponding Interrupt
        Clear the interrupt flag for MAT channel 0 event
```

Algorithm 11 TIMER3_IRQHandler

```
procedure INTERRUPT                                              ▷
    if ultrasonicSensorEdgeCount == 0 then
        Store the rising time into ultrasonicSensorRisingTime variable
    if ultrasonicSensorEdgeCount == 1 then
        if ultrasonicSensorDistance < OBSTACLE_DISTANCE then
            LED_Adjuster(Backward_LED)
            MOTOR_Direction(Backward)
            GoBack
        if ultrasonicSensorDistance > OBSTACLE_DISTANCE then
            LEDAdjusterForwardLED
            MOTORDirectionForward
            GoBack
    ultrasonicSensorEdgeCount++
```

Algorithm 12 ESP8266

procedure SENDCOMMAND(*command*) ▷
  for index < ESP8266BufferSize **do**
    $esp8266Response[index] \leftarrow 0$
  $esp8266ResponseStartIndex \leftarrow esp8266CurrentBufferIndex$
  $ESP8266\_Write(command)$
procedure WAITRESPONSEEND(*command*) ▷
  for ;; **do**
    for esp8266ResponseCurrentIndex < responseEndIndex **do**
      $bufferIndex = StartIndex + CurrentIndex$
      $esp8266Response[CurrentIndex] = Buffer[bufferIndex]$
procedure READRESPONSE Comment
  $esp8266CurrentBufferIndex \leftarrow 0$
  for index < ESP8266BufferSize **do**
    $esp8266Buffer[index] \leftarrow 0$
  while ESP8266_UART-LSR **do**
    if data != 'n' **then**
      $esp8266Buffer[esp8266CurrentBufferIndex] = data$
      $esp8266CurrentBufferIndex{+}{+}$
procedure READDATA ▷
  if data != 'n' **then**
    $esp8266Buffer[esp8266CurrentBufferIndex] = data$
    $esp8266CurrentBufferIndex{+}{+}$

Algorithm 13 PID

procedure PID(*error*) ▷
  $combinedPIDValues \leftarrow 0$
  $combinedPIDValues \leftarrow Kp * error$
  $total\_error \leftarrow total\_error * 3/4 + error * 10$
  $combinedPIDValues \leftarrow combinedPIDValues + total\_error * Ki$
  $combinedPIDValues \leftarrow combinedPIDValues + (Kd * (error - prev_error))/10$
  $prev\_error \leftarrow error$

Algorithm 14 WIFI CHECK

procedure WIFI CHECK ▷
  Start TCP Connection
  Get Information from WIFI
  Change to ROBOT-MODE or start robot

Algorithm 15 Change Start Mode

procedure CHANGE START MODE(*state*) $\triangleright$
    $StartMode \leftarrow state$
    $StopMotorsandLEDs$
    if COMM_TYPE == WIFI_COMM **then**
        Send Information about state via wifi