Cmpe 443 Car Project Report Group Name: Gitt

Serkan Özel(Team Leader) Bahadır Hocamoğlu İsmet Dağlı

Sadullah Gültekin

January 4, 2020

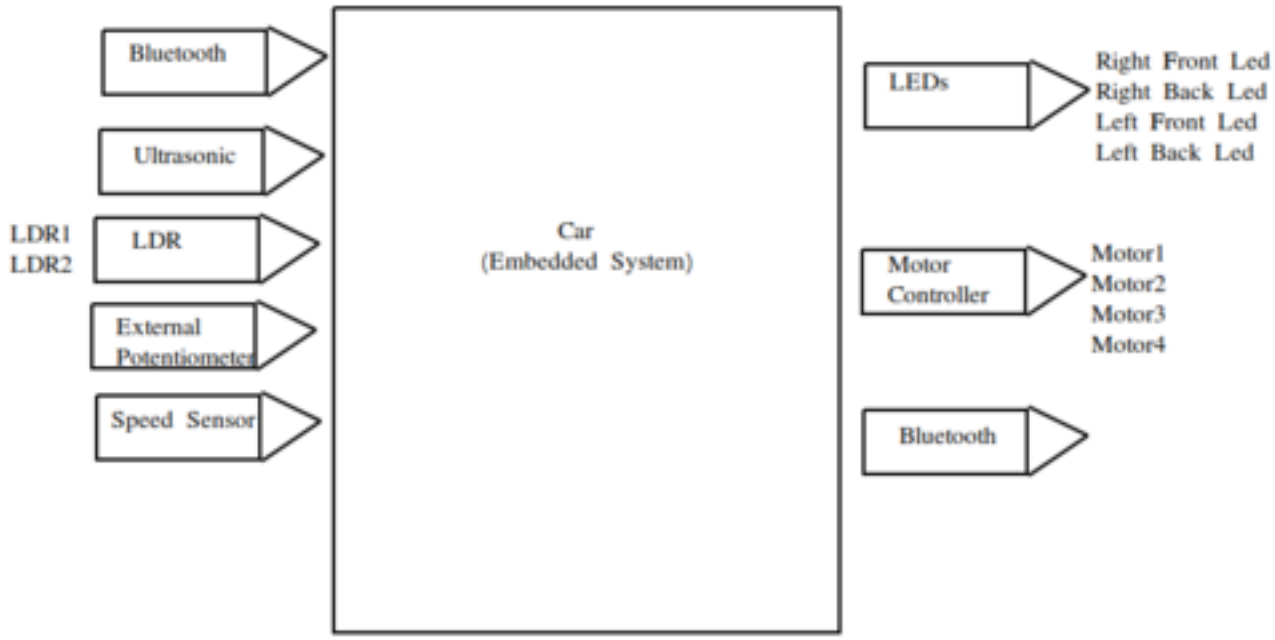
Contents

1 Summary

Project Gitt is an autonomous car that can be controlled remotely over Bluetooth or set to track a wall without too wide turns along the way. It's developed on LPC4088FET208 microcontroller and has various other modules connected to it. The test mode of the car has lots of features like going forward and back, turning left and right, reporting current status of some of the data it's fed by its modules, etc. Further details of the available modes can be found the corresponding section.

The report includes detailed description of connected modules, explanations of how they all come together, the connections between modules and the microcontroller board, and functional logic of the embedded software. The software code embedded in the hardware is supplied with that report separately.

2 Block Diagram



1. Motor Controller Motor controller connects 4 motors to the our main driver. We give 5V and GND to give

power to the motor controller. We connected to output two types for 4 wheels, the first one is for front and back right wheels and the second one is for front and back lefts wheels. They stops initially.

use IN1 and IN2 to move the right front wheel of the car, adjust the speed by changing the ENA.

use IN1 and IN2 to move the right back wheel of the car, adjust the speed by changing the ENA.

use IN3 and IN4 to move the left front wheel of the car, adjust the speed by changing the ENB.

use IN3 and IN4 to move the left back wheel of the car, adjust the speed by changing the ENB.

2. Bluetooth module The Bluetooth module is used to communicate with the android device. First, the connection is established. Then, for each type of movement-command in the car, the car sends a message to the device. In the case of bluetooth connection can be broken or the problems that the device cannot find the car, refresh the page and try to re-establish again.

3. Uart The uart communication is done as a back-up plan for communication if we face with some problems in bluetooth. The uart can be used to give a signal in order to move the vehicle any direction like bluetooth.

4. Light Emitting Diode(LED)

   The LEDs are used to give extra signal about the movement type of the wheel. They are turned of initially.

   turns on front LEDs while moving forward.

   turns on back LEDs while moving backward.

   blinks right LEDs while turning right(clockwise direction).

   blinks left LEDs while turning left(counter-clockwise direction).

5. Ultrasonic This sensor is used to detect the closest object to our vehicle in the right side of the car. When the car detects an obstacle close than 15 cm, the car goes toward opposite direction. When the car detects an obstacle far away than 35 cm, then the car moves

6. Light Dependent Resistor(LDR)

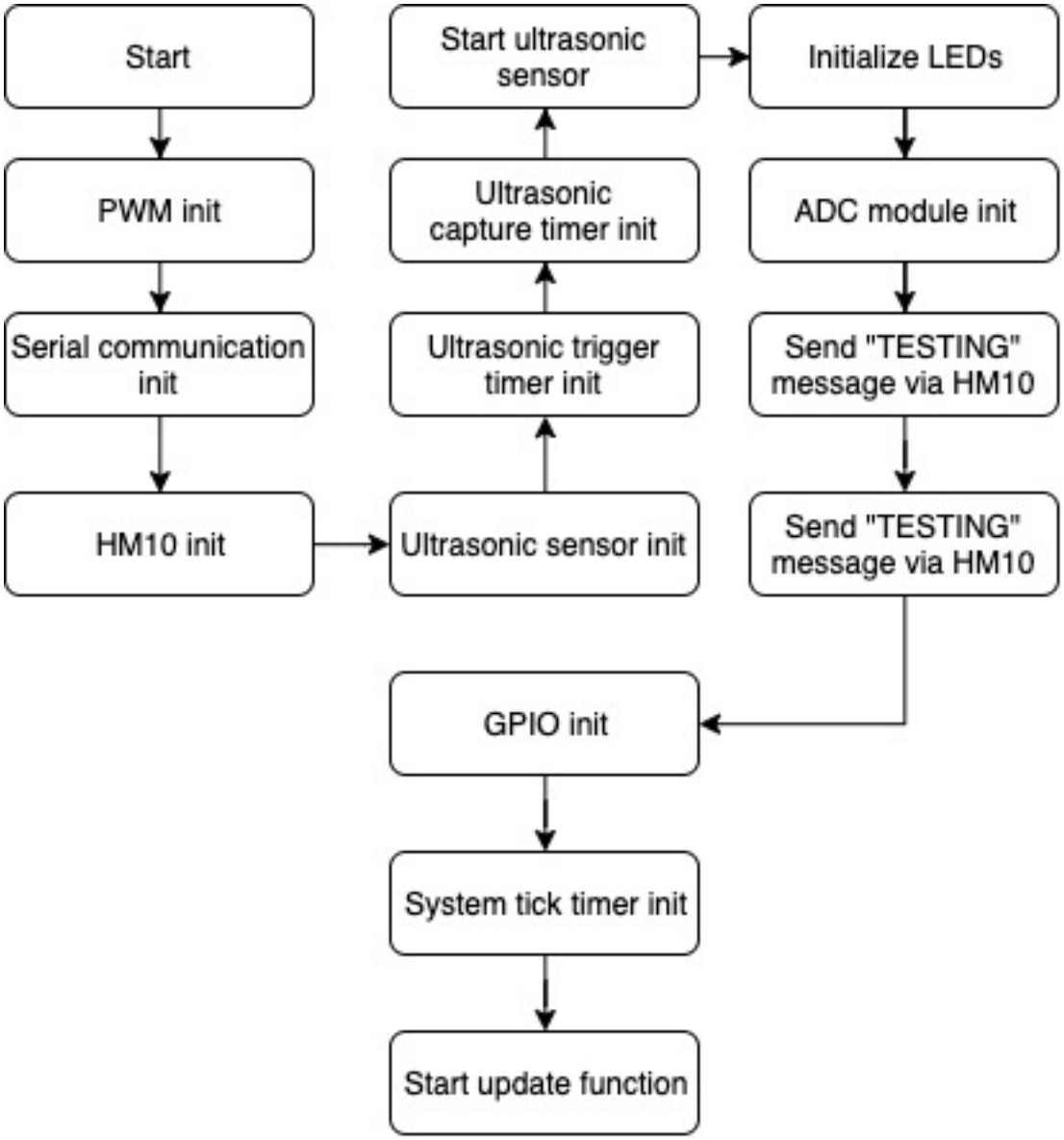
   Two different LDRs are used in the right front corner and the left front corner of the car. There are light sources in the grid and these sensors calculates the light level at the front side of the car. If one of those sensors detects a light level (approximately higher than 300 lumen), then the car stops immediately.

7. External Potentiometer

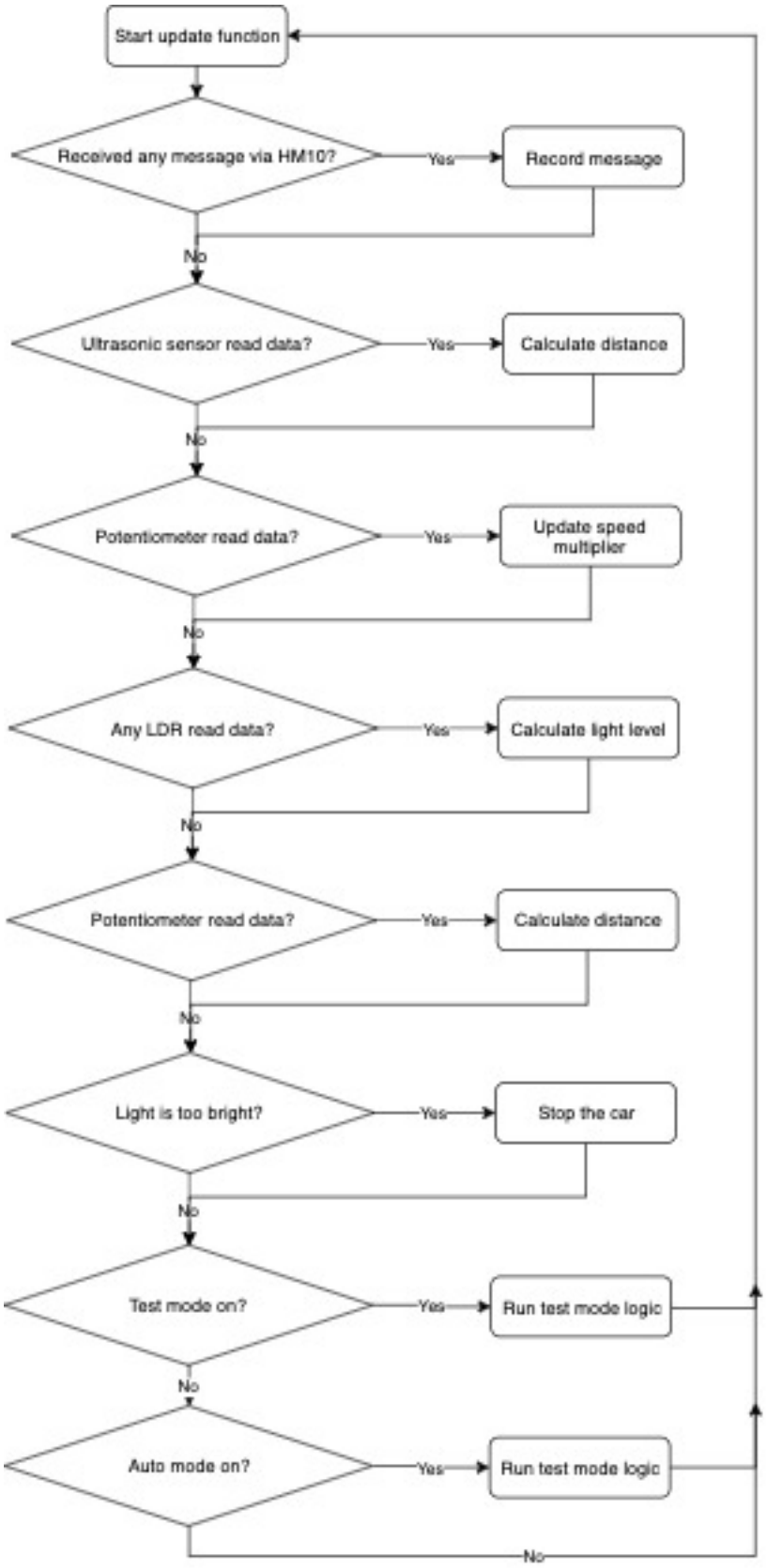
The External Potentiometer is given as a sensor which is connected to the device. We can adjust the speed of the motors by increasing or decreasing the potentiometer.

3 Flowcharts

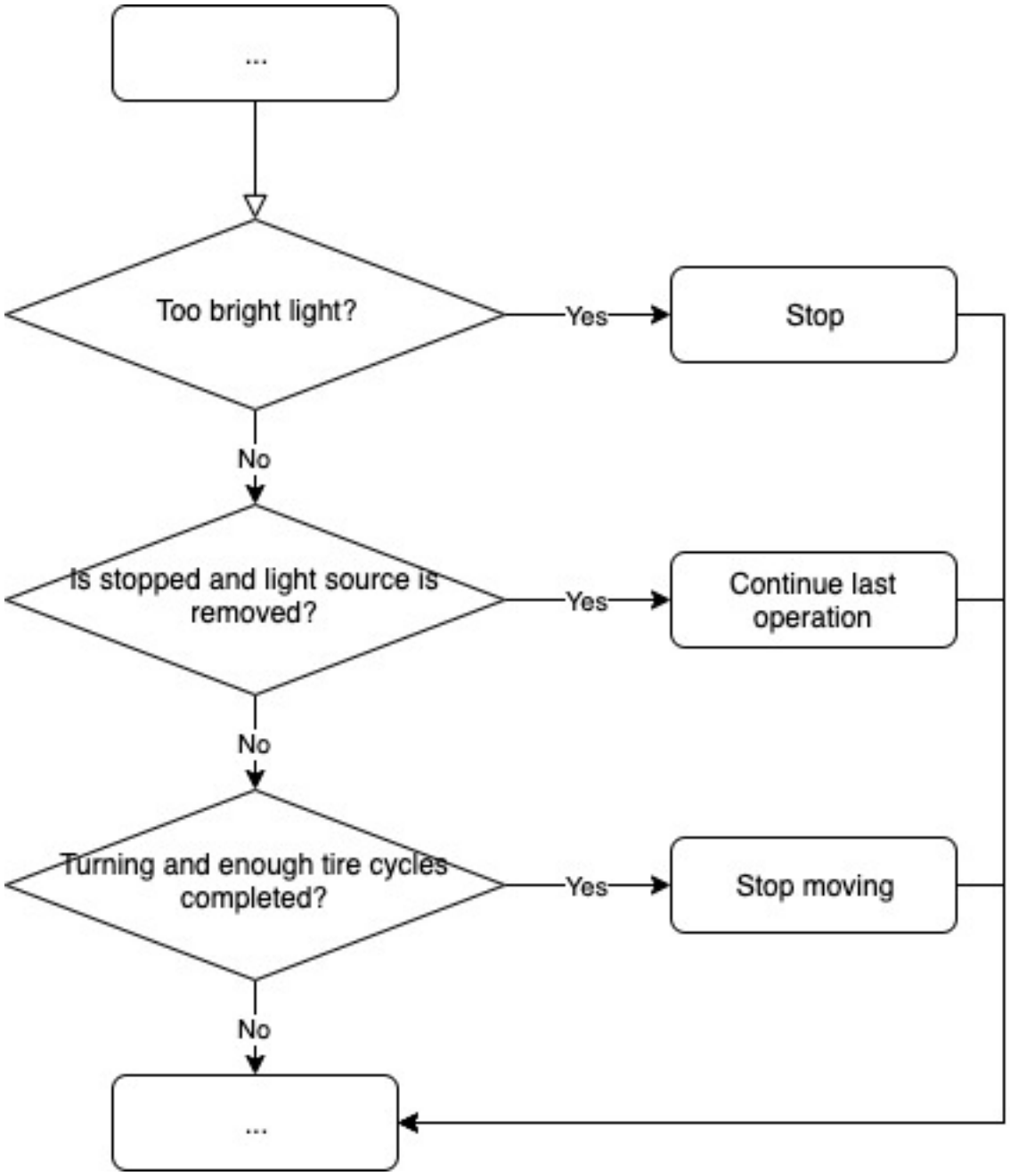
3.1 Main and Init Functions



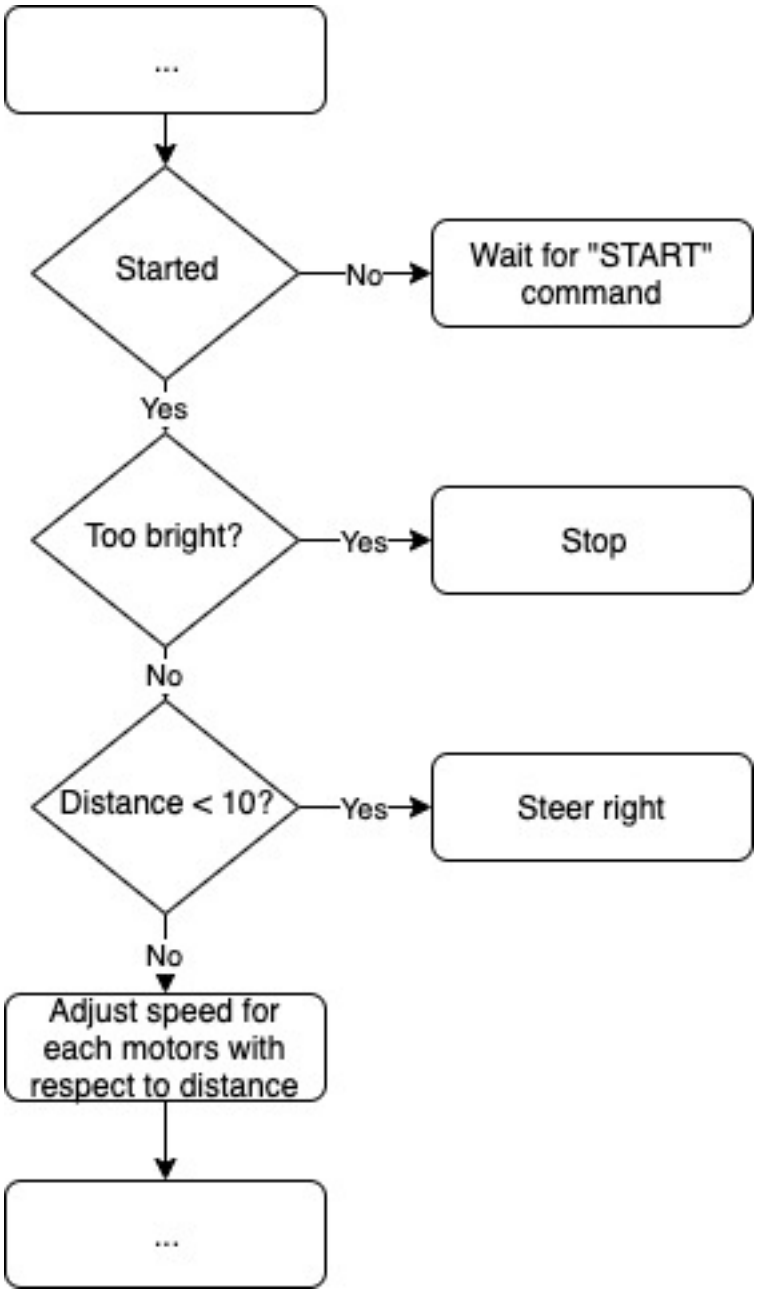
3.2 Update Function



3.3 Test Mode Logic



3.4 Autonomous Mode Logic

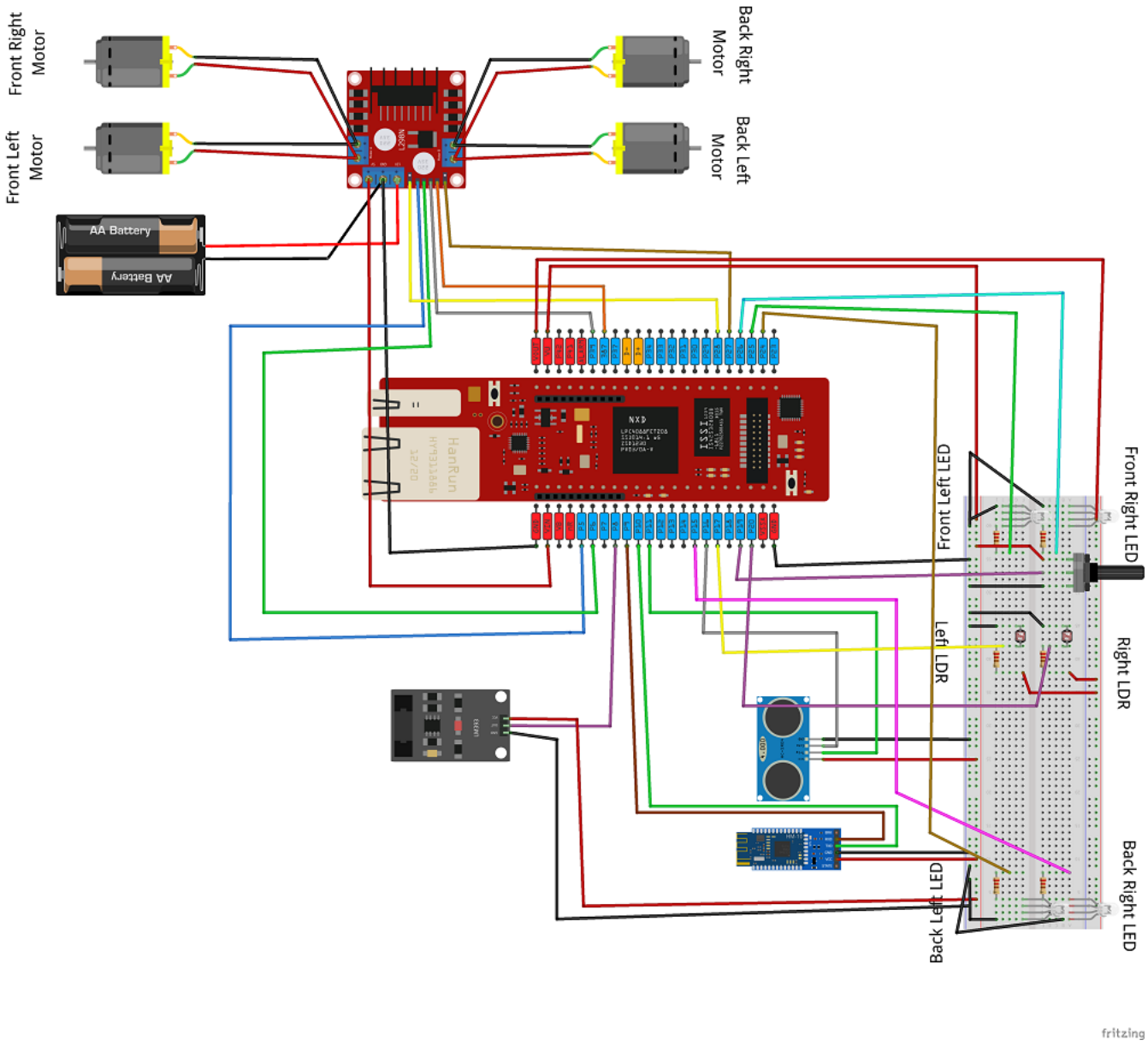


[2] See References for the tool to draw these charts

4 Pin Connection Table

Pin on Module	Pin on Microcontroller
Speed sensor	P8
Ultrasonic sensor trigger	P11
Ultrasonic sensor echo	P16
HM10 TX	P10
HM10 RX	P9
LDR-left analog input	P17
LDR-right analog input	P20
Potentiometer analig input	P19
Motor controller IN1	P5
Motor controller IN2	P6
Motor controller EN1	P28
Motor controller IN3	P39
Motor controller IN4	P38
Motor controller EN2	P27
LED(rear-right)	P15
LED(front-right)	P26
LED(front-left)	P25
LED(rear-left)	P24

5 Circuit Schematics



[1] See References for the tool to draw this

6 How To Use Our System

- Connect battery to the car and give power the motor(switch at the front of the car).
- Connect the car with Bluetooth via android devices by choosing "sado" name.
- When click Status button, the car gives information about the LDR sensors , Ultrasonic sensor and the mode information. The Status button works at both mode.
- While using any movement method at both mode, you can switch the speed with potentiometer at the top of the car.
- The car starts in test mode but at first there is no movement on motors.
- Click any movement button like forward,backward, left or right. The cars sends a specific movement message to the android device.
  - If click left, sends "LEFT" command to device and turns 90 degree to left. The LEDs at the left blink twice a second. While turning, if any extra button is clicked, first completes 90 degree that direction, than executes the second command.
  - If click left, sends "RIGHT" command to device and turns 90 degree to right. The LEDs at the right blink twice a second. While turning, if any extra button is clicked, first completes 90 degree that direction, than executes the second command.
  - If click forward, sends a "FORWARD" command to device. Only the LEDs at the front are turned on. You can change movement to any direction if there is any extra command.
  - If click backward, sends a "BACKWARD" command to device. Only the LEDs at the backward are turned on. You can change movement to any direction if there is any extra command.
- After testing the movement, click Auto mode to test. The car sends a "Autonomous" message to the android. Firstly, it waits until clicking START button.
- Click START button in the auto mode, the car keeps the distance along the wall at a certain level.
- Until clicking "TEST" button or detecting a light more than a certain level, the car continues to move.

- Clicking "Test button, the car stops immediately and enters to "TESTING" mode. We can continue in test mode for testing.
- Detecting high light level, the cars stops immediately and sends a FINISH message to Android.

7 Detailed Description of Functional Parts

7.1 Modes

The car has two operation modes: Test mode and Autonomous mode that can be changed via bluetooth application. Regardless of the operation mode, the car has these global behaviours:

- When device stops, all leds are turned off.
- When device is moving forward, leds on the front are turned on.
- When device is moving back, leds on the back are turned on.
- When device is turning left, leds on the left are blinking twice a second.
- When device is turning right, leds on the right are blinking twice a second.
- When device get "STATUS" string from serial communication it sends its status as response.
- An external potentiometer is used to adjust speed of the car.
- The car echoes back every commands it receives.

7.1.1 Test Mode

This mode is used to test the car. Testing functionalities include turning left and right, going back and forward, and light sensors. The distance sensor value can be learned via communication, via bluetooth, android application.

- The car sends TESTING when entered to test mode.
- The car turns 90 degree left and stop after getting LEFT from serial communication.
- The car turns 90 degree right and stop after getting RIGHT from serial communication.
- The car starts moving forward after getting FORWARD from serial communication.

- The car starts moving back after getting BACK from serial communication.
- The car stops after getting STOP from serial communication.
- The car stops under light and continues what it was doing before after light goes.
- The car switches to autonomous mode after getting AUTO from serial communication.

7.1.2 Autonomous Mode

The car's task in this mode is to follow a wall that is on the left side of the car reasonable distance away, and stop before passing a LED strip via LDRs.

- At the beginning of autonomous mode, the car says AUTONOMOUS.
- After entering this mode, the car stays idle until it receives START command.
- While moving and next to the wall, the car moves along the wall, within 15 to 35 centimeters from the wall
- After seeing the LED strip the car stops and says FINISH.
- After getting TEST command in autonomous mode, the car switches to test mode.

7.2 Pseudocode

The Pseudocode is provided in this part.In the code, first we initialized the whole different parts of the car. Then the code explains how the car is working

function init():

PWMInit()

SerialInit()

HM10Init()

UltrasonicInit()

UltrasonicTriggerTimerInit()

UltrasonicCaptureTimerInit()

UltrasonicStartTriggerTimer()

LED1Init()

LED2Init()

LED3Init();

LED4Init();

ADCInit();

// Initialize status objects status.distance = 0

status.lightLevelLeft = 0

status.lightLevelRight = 0

status.opmode = "TEST"

status.underLight = 0

status.currentOperation = STOP // At the beginning car should stop

status.willContinue = 0

status.wheelToothCount = 0

CarLEDs_stop() // Adjust leds according to stop operation

ChangeMotor1Speed(100) // These do not move car just adjusts speed scaling to 100, meaning, potentiometer is not scaled in test mode

ChangeMotor2Speed(100)

HM10SendCommand("TESTING") // Send TESTING at the beginning since TEST mode is entered at first.

GPIOInit()

SysTickInit()

function update():

if(serialNewDataAvailable): // If the car got new data via UART(e.g putty)

serialNewDataAvailable = 0;

if(SerialResponseReceived()): // if last received character is r

SerialSendCRLN()

SerialSendString(serialReceived)

else

strncat(serialReceived, &serialReceivedCharacter, 1); // If not append to form complete

response

```
if(HM10NewDataAvailable): // If the car got new data via bluetooth
    HM10NewDataAvailable = 0
    if(HM10_ResponseReceived()): // if last received character is \n, the command is finished, start
processing it
        HM10ProcessResponse()
        HM10ClearBuffer()
if(ultrasonicSensorNewDataAvailable): // If ultrasonic data is avaliable
    ultrasonicSensorNewDataAvailable = 0
    status.distance = (ultrasonicSensorFallingCaptureTime - ultrasonicSensorRisingCaptureTime)/58;
if(ADCNewSpeedAvailable): // If new speed from potentiometer is available
    temp = (ADCGetLastSpeed()*100) / 4095;
    if (temp< 5): // To reduce noise and problem we faced crop 0-5 interval to 0.
        status.speed = 0
    else :
        status.speed = temp // otherwise write the same
if(ADCNewLeftLightAvailable): // If new left light level is available
    status.lightLevelLeft = 1023 - (ADCGetLastLeftLight() / 4)
if(ADCNewRightLightAvailable): // If new right light level is available
    status.lightLevelRight = 1023 - (ADCGetLastRightLight() / 4);
// If light level is high on either side, set the variable
if(status.lightLevelLeft > LIGHTTHRESHOLD or status.lightLevelRight > LIGHTTHRESHOLD)
    status.underLight = 1
else:
    status.underLight = 0
if(strcmp(status.opmode,"TEST") equals 0):
    ChangeMotor1Speed(100); // Do not apply scaling to potentiometer value
    ChangeMotor2Speed(100);
```

```
// Stop if moving forward or backward, due to 90 degree turn
if(((status.currentOperation equals FORWARD) or (status.currentOperation equals BACKWARD))
and status.underLight)): // Stop when light is on you
    lastOperation = status.currentOperation
    status.currentOperation = STOP
    StopMotors()
    status.willContinue = 1
// If the car is stopping, and stopped due to light, continue last operation
if(status.currentOperation == STOP && status.willContinue && status.underLight == 0)
    status.currentOperation = lastOperation;
    // according to last operation change leds and motor directions
    if(status.currentOperation == FORWARD):
        CarLEDsGoingForward()
        MoveForward()
    else if(status.currentOperation == BACKWARD):
        CarLEDsGoingBackward()
        MoveBackward()
    else if(status.currentOperation == STOP):
        CarLEDsStop()
        StopMotors()
    status.willContinue = 0 // willContinue should be 1 in stop mode and due to light is stop
// Turning left or right wheelToothCount is reset, stop if we reach the threshold for 90 degree turn
if((status.currentOperation equals LEFT or status.currentOperation or RIGHT)
    and status.wheelToothCount BiggerThan TURNCOUNTFOR90)
    status.wheelToothCount = 0
    status.currentOperation = STOP
    StopMotors()
    CarLEDsStop()
```

```
else if(strcmp(status.opmode, "AUTO") equals 0)// The code runs in auto mode
    if(status.started == 1): // Only move if started
        // Clip distance to 40 maximum to avoid our speed functions' overflow situation
        status.distance = status.distance < 40 ? status.distance : 40
        // If underlight stop and respond FINISH
        if(status.underLight == 1):
            status.started = 0
            HM10SendCommand("FINISH")
            HM10SendCRLN()
        else if (status.distance lowerThan 10): // If too close turn right
            ChangeMotor2Speed(10)
            ChangeMotor1Speed(70)
            CarLEDsTurningRight()
        // Else depending on the distance adjust turn
        else:
            tempSpeed = (uint32_t) (8.0 + (75.0 / (1 + pow(2.718281828459, ((5.0 - status.distance) / 3.0)))))
            ChangeMotor2Speed(tempSpeed)
            ChangeMotor1Speed(91 - tempSpeed)
            if(temp_speed lowerThan (91 - tempSpeed)) :
                CarLEDsTurningRight()
            else :
                CarLEDsTurningLeft()
    else if(status.started equals 0): // If not started stop
        status.currentOperation = STOP
        StopMotors()
        status.willContinue = 0
        CarLEDsStop()
```

```
function MoveForward()
        Motor1ChangeDirection(CLOCKWISE)
        Motor2ChangeDirection(CLOCKWISE)
function MoveBackward():
        Motor1ChangeDirection(COUNTERCLOCKWISE)
        Motor2ChangeDirection(COUNTERCLOCKWISE)
function TurnLeft():
        Motor1ChangeDirection(COUNTERCLOCKWISE)
        Motor2ChangeDirection(CLOCKWISE)
function void StopMotors()
        Motor1ChangeDirection(STOPMOVING)
        Motor2ChangeDirection(STOPMOVING)
function ChangeMotor1Speed(uint32t speed)
        if(speed > 100)
                speed = 100
        PWMWriteMotor1(100-(speed*status.speed /100))
function PWMWriteMotor1(uint32t TimeON)
        int temp
        if(TimeON > 100)
                TimeON = 100
        temp = PWMX->MR0 * (100-TimeON)/100
        PWMX->MR3 = temp
        //Enable PWM Match Register Latch.
        PWMX->LER |= 1 « 3
function CarLEDsStop():
        status.LED1Status = OFF
        status.LED2Status = OFF
        status.LED3Status = OFF
```

```
        status.LED4Status = OFF
    function CarLEDsGoingForward():
        status.LED1Status = OFF
        status.LED2Status = ON
        status.LED3Status = ON
        status.LED4Status = OFF
    function CarLEDsTurningLeft():
        status.LED1Status = OFF
        status.LED2Status = OFF
        // if check needed for autonomous mode, calls this function causes leds to always be on
        if(status.LED3Status NotEqual BLINKON AND status.LED3Status NotEqual BLINKOFF)
            status.LED3Status = BLINKON
        if(status.LED4Status NotEqual BLINKON AND status.LED4Status NotEqual BLINKOFF)
            status.LED4Status = BLINKON
    function Serial_SendCRLN():
        serialTransmitData = "\r\n"
        SerialSendData()
    function SerialSendString(char * data):
        serialTransmitData = data
        SerialSendData()
```

7.3 Data Structures

7.3.1 Global Status Object

We have a global status struct that is defined inside Datastructures.h and declared inside main.c. We share this status struct with every needed file so that the car has a single status that can be accesible everywhere. The status object includes the following:

Status

- uint32 distance: distance from ultrasonic sensor

- uint16 lightLevelLeft: light level measured with left ldr, between 0-1024
- uint16 lightLevelRight: light level measured with right ldr, between 0-1024
- char* opmode: Operation mode that is either AUTO or TEST
- uint32 speed: Current speed between 0-100, read via potentiometer
- Operations currentOperation: A struct that defines what the car is doing right now
- uint8 underLight: A 0/1 variable that indicates if the car is underlight or not. Underlight threshold is 280.
- uint8 willContinue: Indicates if the car is stopped due to light. If this is true last operation in main.c is put into currentOperation.
- uint8 started: Indicates if the car is started in autonomous mode, until then,
- uint32 wheelToothCount: Every wheel tooth increases this by 1, we have a threshold that is defined in datastructures.h that defines after how many teeth the car should stop while turning left or right, the threshold is currently 5.
- LEDSTATUS LED1Status: Indicates LED1 status
- LEDSTATUS LED2Status
- LEDSTATUS LED3Status
- LEDSTATUS LED4Status

7.3.2 Operations Enum

This enum holds the current operations for the test mode of the device. The possible values are:

- FORWARD
- BACKWARD
- RIGHT
- LEFT
- STOP

7.3.3 LedStatus Enum

This enum holds the current status of a LED. The possible values are. Update leds function in systick interrupt handler updates the leds according to their state. The states can be one of the following:

- BLINKON -> next state is BLINKOFF
- BLINKOFF -> next state is BLINKON
- ON -> next state is ON
- OFF -> next state is OFF

7.3.4 MotorDirection Enum

This enum holds the current direction of a motor. The possible values are:

- STOPMOVING
- COUNTERCLOCKWISE
- CLOCKWISE

8 Expense List

In our project, we didn't buy any extra equipment. So, we don't have an additional expense.

9 References

1. www.fritzing.com
2. www.draw.io