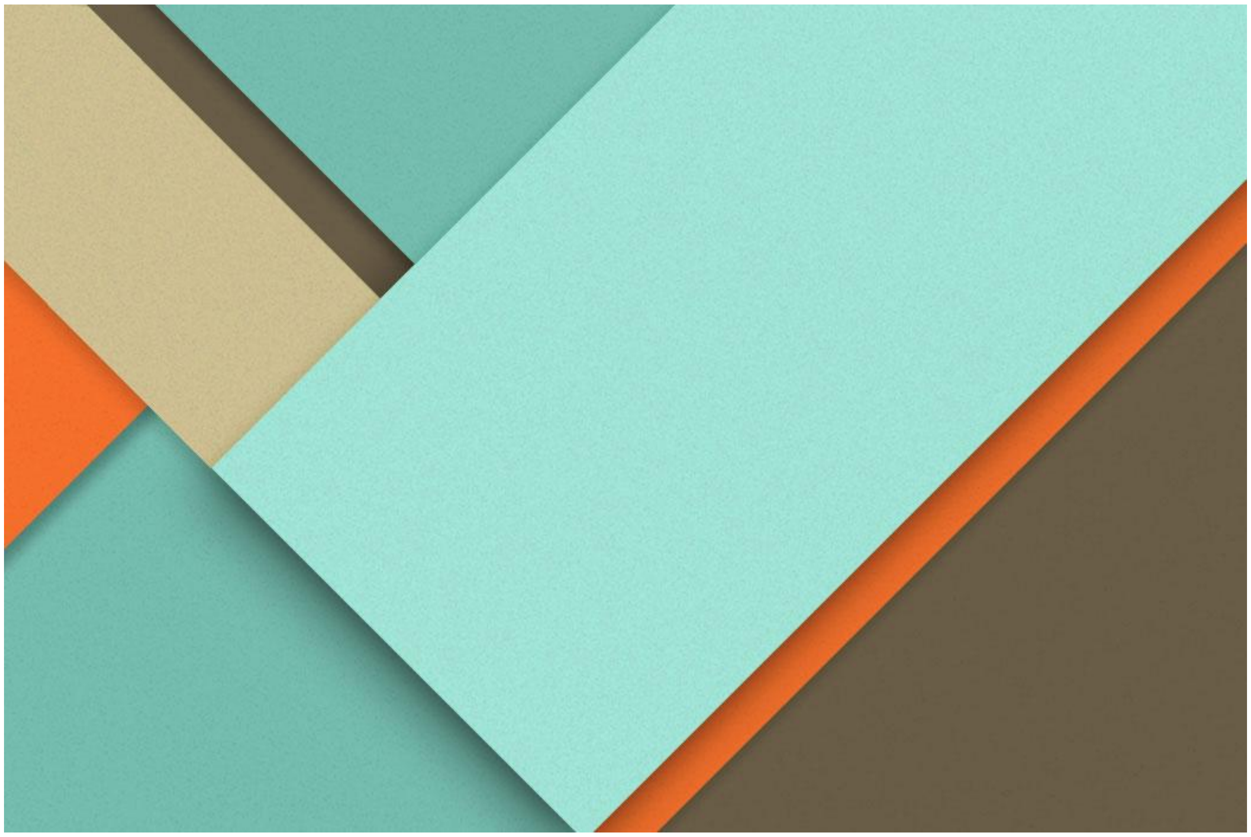


Team 17

Name	ID	Tutorial Group
Mark Sroor	46-5861	T-22
Lougina Hatem	46-13014	T-22
Dareen Mohamed	46-0416	T-18
Omar Galal	46-1097	T-12

Project Idea

The main idea is to implement a micro-controller based car with multiple safety and entertainment features .

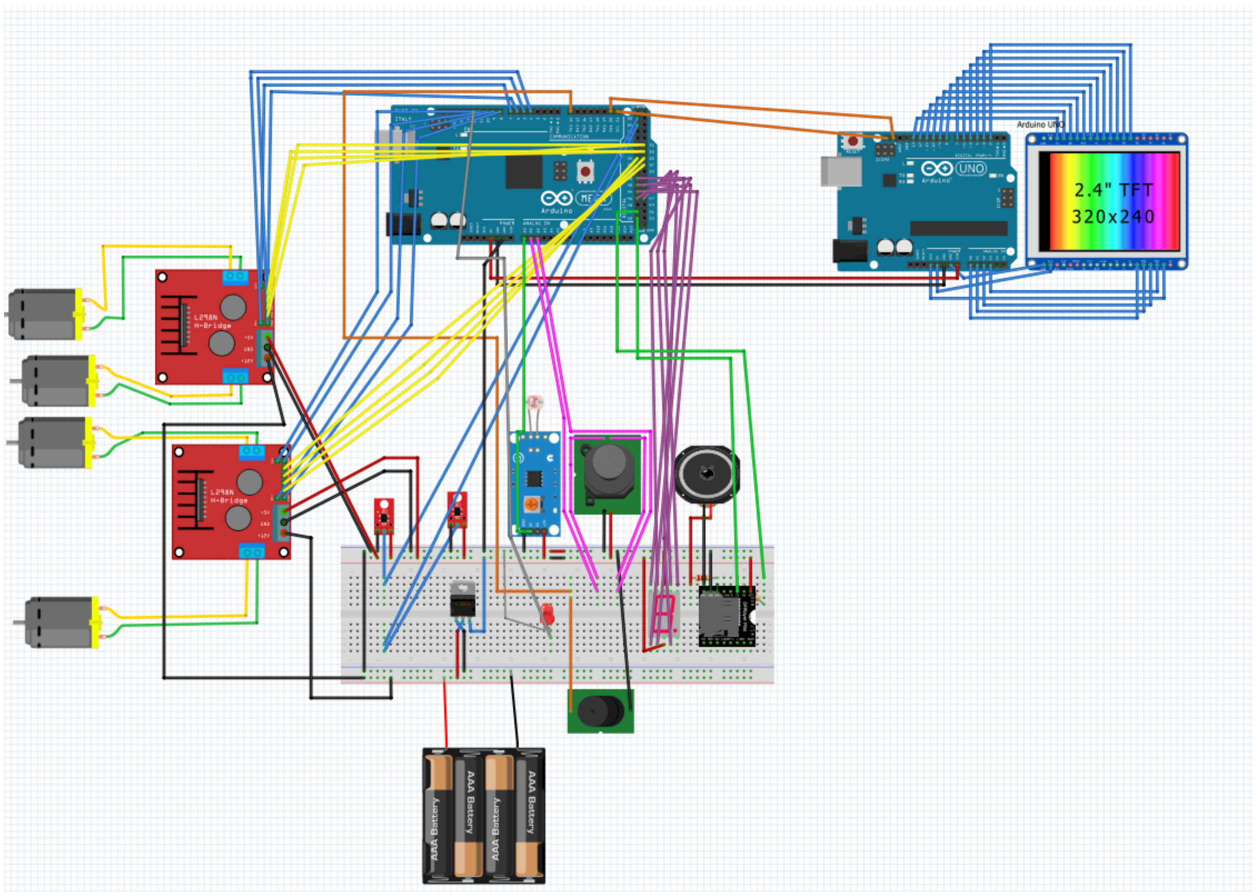
and apply the scheduling techniques that raise the safety level to a maximum ,

It includes an autonomous lane departure system to avoid sway while moving and keeping the driving path to avoid collision; it corrects the path with the right speed while alerting the driver. In addition, the current gear of the car is displayed and can be changed using a joystick. Moreover, adaptive headlights safety is not just limited to the driver but to those around him as well. The headlights are adjusted according to the brightness surrounding the car, meaning that at night the headlights will work with enough intensity to show the road but not high enough to obstruct the vision of other drivers. Finally, the sound system, user can play any mp3 file through the touchscreen. The mp3 file name will be displayed and the user can Play/Pause and iterate between the saved mp3 files.

Approach

First we worked on getting every component to be functional separately. Then we combined them together. After that we combined both arduinos and integrated everything.

Circuit


Arduino UNO
UNO
Arduino
2.4" TFT
320x240
L298N H-Bridge
L298N H-Bridge
AAA Battery
AAA Battery
AAA Battery
AAA Battery

Handling Inputs:

Regarding our inputs, they were from our sensors like our light detection sensor, lane detection sensor. These sensors were connected to the pwm ports of our arduino as we needed from analog to digital.

Handling Outputs:

Regarding our outputs, they were from our sensors like our light detection sensor, lane detection sensor. These sensors were connected to the pwm ports of our arduino as we needed from analog to digital.

Input and output pins and their configurations:

Arduino Mega Pins Input/Output:

- left (Left Lane Sensor), Pin =22, Type = analog input
- right (Right Lane Sensor), Pin = 24, Type analog input
-
- led1, Pin = 12, Type = digital output

Motor Pins

- const int IN1 = 30; //right back wheel Forward
- const int IN2 = 31; //right back wheel Reverse
- const int IN3 = 32; //right front wheel Forward
- const int IN4 = 33; //right front wheel Reverse
- //To turn motor on, off and control speed
- int en3 = 6; //enable for right back wheel pwm
- int en4 = 7; //enable for right back wheel pwm
- int en2 = 4; //enable for right front wheel pwm
- int en1 = 5; //enable for right front wheel pwm
-
- const int IN11 = 34; //left front wheel Forward
- const int IN22 = 35; //left front wheel Reverse
- const int IN33 = 36; //left back wheel Forward
- const int IN44 = 37; //left back wheel Reverse
- //To turn motor on, off and control speed
- int en33 = 11; // enable for left back wheel
- int en44 = 10; // enable for left back wheel

- int en22 = 9;   // enable for left front wheel
- int en11 = 8;   // enable for left front wheel
-

VRx (Gear Input Direction), Pin = A1, Type = analog input

VRy (Gear Input Direction), Pin = A2, Type = analog input

SW (Gear Input Direction of Parking), Pin = 53, Type = digital input

a, Pin = 24, Type = digital output

3.5. Name = b, Pin = 22, Type = digital output

3.6. Name = c, Pin = 16, Type = digital output

3.7. Name = d, Pin = 15, Type = digital output

3.8. Name = e, Pin = 14, Type = digital output

3.9. Name = f, Pin = 19, Type = digital output

3.10. Name = g, Pin = 18, Type = digital output

3.11. Name = h, Pin = 17, Type = digital output

Arduino Uno Pins Input/Output:

1. Touch Screen Pins 1.1. Name = YP, Pin = A3, Type = analog input

1.2. Name = XM, Pin = A2, Type = analog input

1.3. Name = YM, Pin = 9, Type = digital input

1.4. Name = XP, Pin = 8, Type = digital input

1.5. Name = LCD_RD, Pin = A0, Type = analog input

1.6. Name = LCD_WR, Pin = A1, Type = analog input

1.7. Name = LCD_CD, Pin = A2, Type = analog input

1.8. Name = LCD_CS, Pin = A3, Type = analog input

1.9. Name = LCD_RESET, Pin = A4, Type = analog input

1.10. Name = LED_PIN, Pin = A5, Type = digital input

Components Used

I. 4WD Smart Robot Car Chassis Kit

We used the 4WD smart robot car chassis kit as the main support structure for ou car. It comes with 4 DC gear motors.

II. Arduino Shield "LCD TFT 2.4 With Touch Screen"

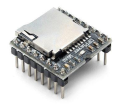
This touch screen acted as the graphical interface for the arduino. It was used to control the sound system of the car, such as pausing/playing a song, playing the previous/next song, and displaying the song name on the screen.

III. 4pin Photoresistor, LDR Module – Optical Sensitive Resistance Light Detection Sensor Module

This sensor was used to detect light intensity.

IV. MP3 Mini Player

Songs were played using the MP3 mini player via a speaker.

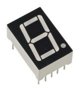


V. Breadboard

Used to connect the components

VI. 7 Segment Display

7 Segment display was used to display the current gear.



VII. Joystick

A joystick was used to control the gear.

VIII. QRE1113 Digital Line Tracker (Follower) Sensor

The Line tracker sensor was used to keep the car moving into its lane and detect if it's leaving the lane.

IX. Arduino Uno

X. Arduino Mega

XI. LED

Used as the car headlights.

Libraries

```
#include <Wire.h>

//This library allows you to communicate with I2C / TWI devices

// Functions:begin(),requestFrom(),beginTransmission(),endTransmission(),
write(),available(),read(),


#include <Arduino_FreeRTOS.h>

//This library provides the arduino with real time scheduling functionality, inter-task
communication, timing and synchronization primitives and allows the use of these
functions

//Functions:
xTaskCreate(),pdMS_TO_TICKS(),xTaskGetTickCount(),vTaskDelayUntil()


#include "SoftwareSerial.h"

//allow serial communication on digital pins of the Arduino other than pins  0 and 1,
using software to replicate the functionality.

//Functions: begin(),write()


#include <Fonts/FreeSans9pt7b.h>

#include <Fonts/FreeSans12pt7b.h>

#include <FreeDefaultFonts.h>
```

```
#include <Adafruit_GFX.h>,#include <MCUFRIEND_kbv.h>
//graphics library used for all LCD displays providing a set of graphics functions
//Functions:width(),height(),readID(),begin(),
fillScreen(),drawFastHLine(),setFont(),setCursor(),setTextColor(),
setTextSize(),print(),fillRect()

#include <TouchScreen.h>
//Functions:getPoint(),
```

Handling Inputs:

Regarding our inputs, they were from our sensors like our slim

FreeRTOS:

Features were divided upon Arduino Uno (master) and Arduino Mega (Slave). Where Arduino Uno was assigned the screen only and the remaining features & functionalities were assigned to the Arduino Mega. The Lane Keeping Assist's feature was given the highest priority (3 but not the only task with this priority) and had a delay of only 100ms.

Lane Keeping Assist (LKA) :

- Controlling the motors' speed ,motors' movement and line sensor were all handled in a periodic task with a priority of 3 and a mere delay of 100ms to prevent starvation of the other tasks.

Control Indicators (CI) :

- Controlling the joystick was put into a task in the Mega that reads the input and sets the 7-Segment Display according to the current gear with priority of 2 and a periodic delay of 1000ms.
- Light intensity control was assigned a task of priority 2 and periodic delay of 750ms which sets the LED according to the light intensity being read.

Sound System (SS) :

- Getting input from the touchscreen and sending output to the slave were handled in the Uno without the use of tasks as there was only one functionality in the Uno. Manipulating the input in the slave was not put in a task.

- Sending the song that is currently being played to the master was put in a task in the Mega with priority equal to 3 with a periodic delay of 1500ms.

Limitations :

A number of limitations and setbacks were faced during the implementation of the project that were later overcome after testing and researching.

- Playing songs from the MP3 module in a timely manner when the LCD gets an input was challenging because of the delays from the I2C , tasks priorities and the fact that the MP3 module repeatedly disconnects.

- Light sensitivity of the LDR was not very accurate.

- Speaker gets burned frequently whenever a flow of high voltage occurs.

- FreeRTOS and giving the tasks different priorities to ensure that no starvation or deadlocks occur to any of the tasks.

- Communication between Arduino Uno (master) and Arduino Mega (slave) using I2C.

- Line Sensor needed to be lined very closely to the lane that was being followed ,which should not be reflective to avoid inaccurate results.

Work Division :

All team members worked collectively on all parts to deliver the requirements and all features were implemented by all members together with no feature assigned exclusively to any member.