Project Report

TEAM 34 (TEAM EMBEDD BY 9:30

OMAR K. YEHIA 49-10704 T-19

Brief Description:

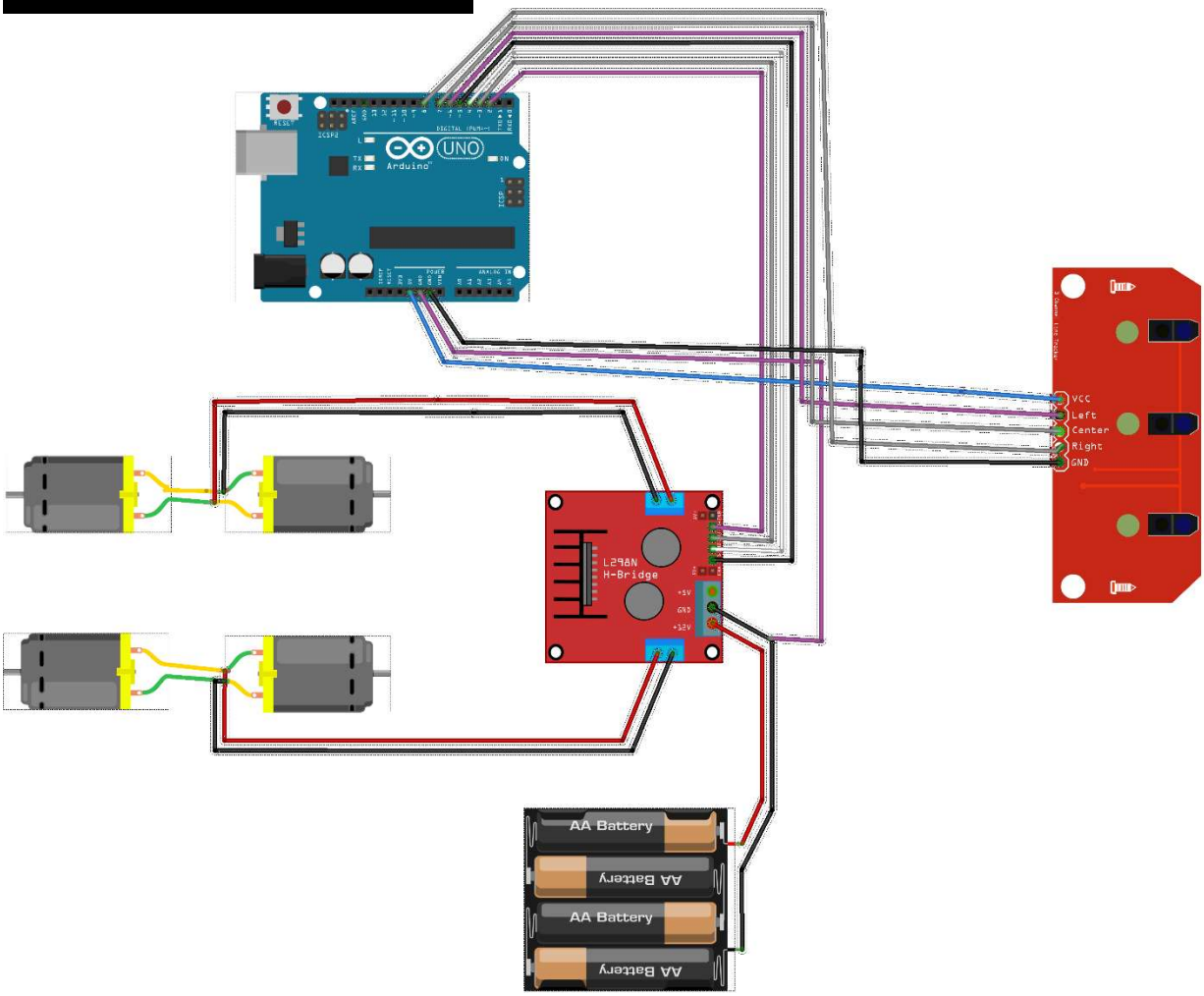
The aim of this project was to use Arduino C to create a car that would be able to keep inside a lane automatically without any use of inputs. The car would also need to have adaptive headlights that reacted to the environment around it and a touch screen controlled MP3 player. Our intuition was to try to have the car be as independent possible from all other modules in order to simulate an approach not unlike the ones used in designing real embedded systems in full sized cars. All other modules in the car were handled as normal tasks that were handled with one microcontroller and were scheduled using FreeRTOS.

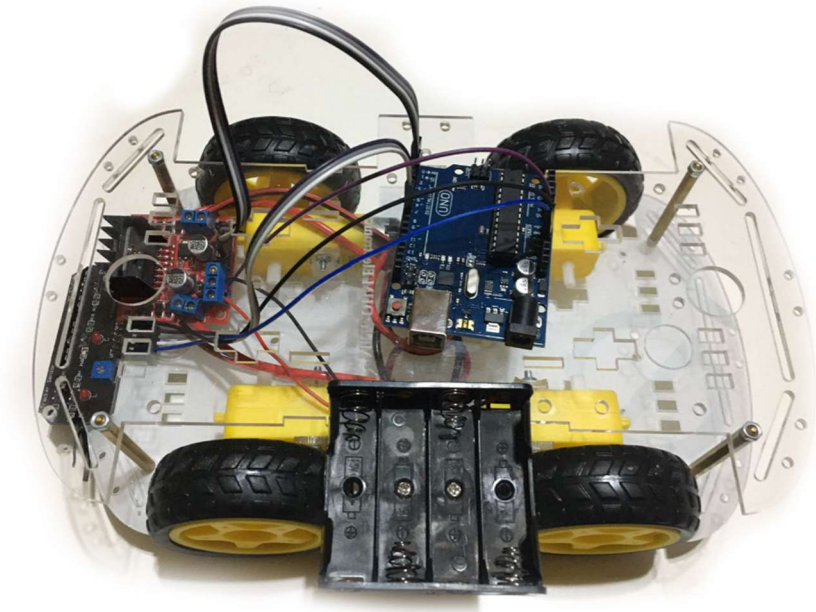
Components Used:

1) Arduino Uno: Used as the microcontroller for the car.
2) Car Chassis and Wheels: Used to house all other components.
3) 4 x DC Motors: Used to move the car.
4) AA Batteries: Used to power the DC Motors.
5) Motor Driver Module (L298): Used to communicate between the Arduino and the car.
6) 3-Channels Path Tracking Module (TCRT5000): Used to follow the line on the ground.
7) Arduino Mega: Used as the microcontroller for all other functionalities.
8) 7-Segment Display: Used to display car gear.
9) Joystick Module: Used to control car gear.
10) Mini MP3 Player Module: Used to read the SD Card and play music on the speaker
11) Mini SD Card: Used to store the songs in the MP3 format.
12) LDR Sensor: used to sense light in the environment.
13) 0.25W 32 Ohm Speaker: Used to play music.
14) 2.4” TFT Touch Screen LCD Shield: Used for control of the MP3 Module functionalities
15) Breadboard: Used to house the circuitry of the modules
16) Male to Male Jumper Wires: Used to connect between the various modules and MCUs
17) Female to Male Jumper Wires: Used to connect between the various modules and MCUs

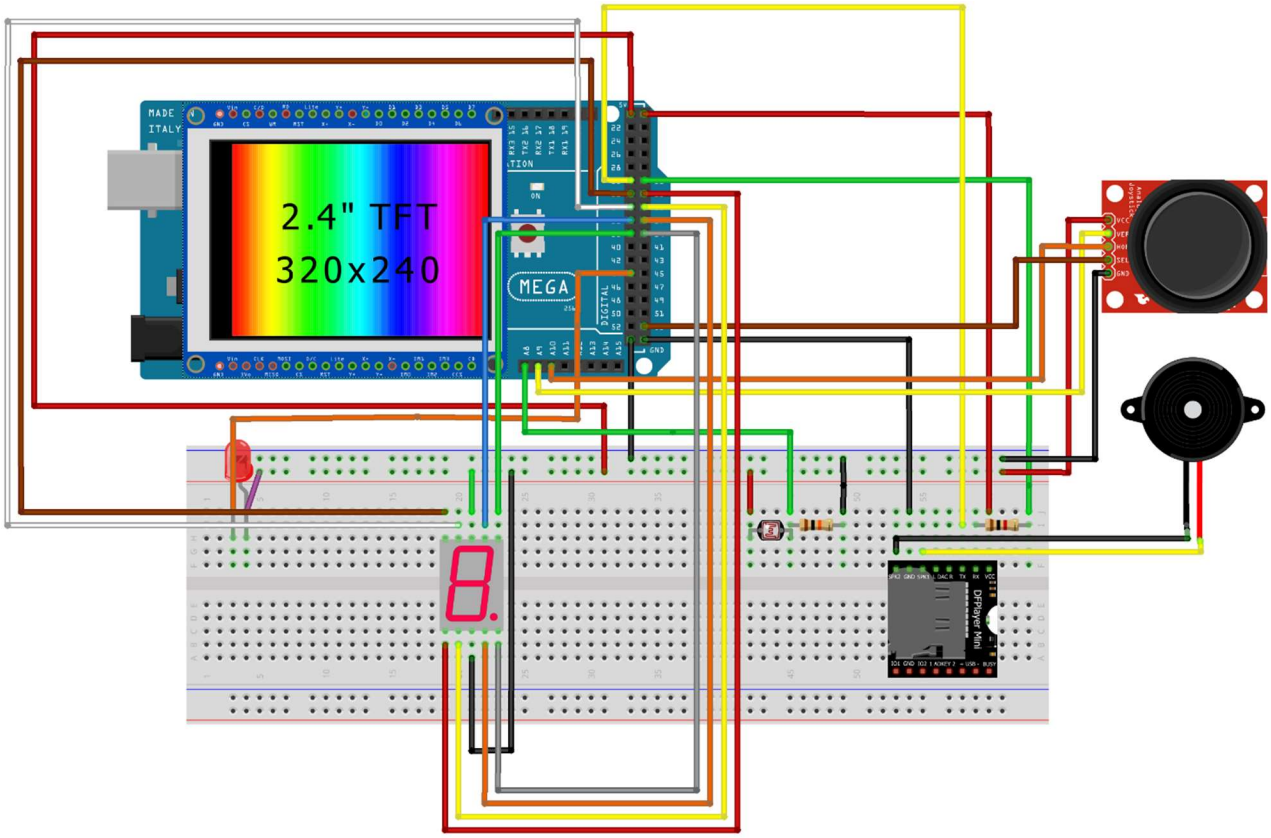
Full Circuit Diagrams

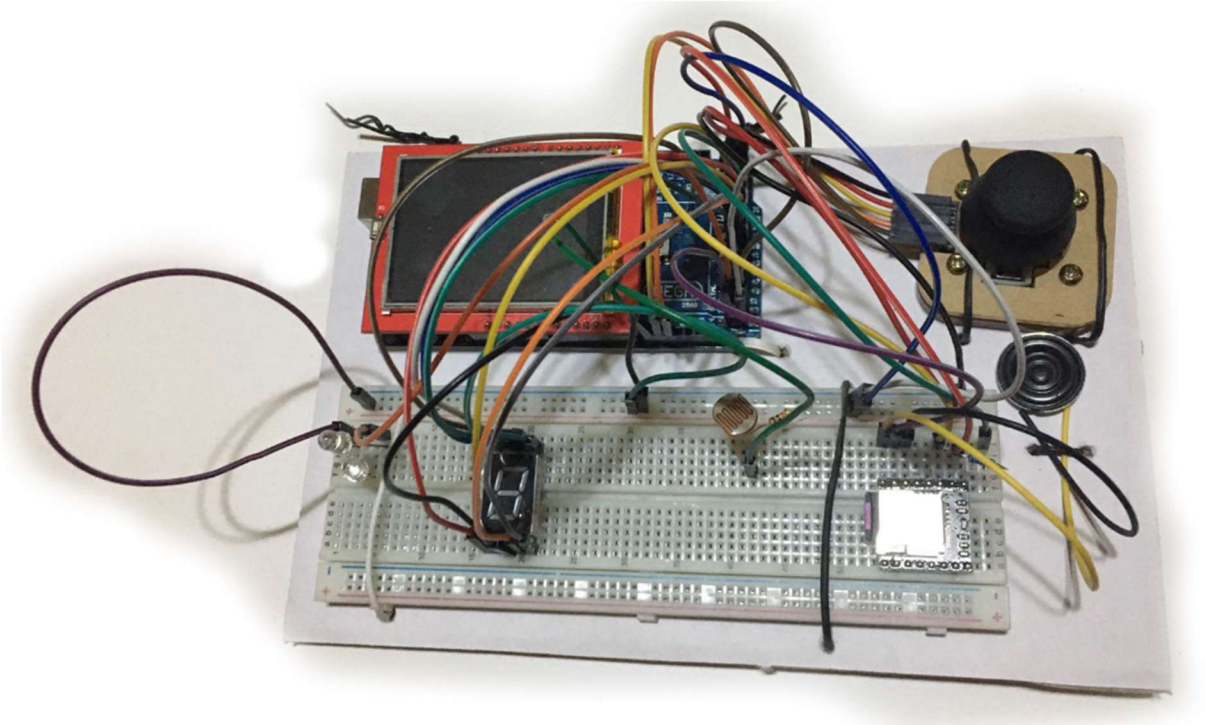
Arduino Uno (Car Functionalities)





Arduino Mega (All Other Functionality)





Libraries and Functions Used:

```
<Arduino_FreeRTOS.h> //For RTOS
TickType_t xLastWakeTime; //For Delaying
const TickType_t xDelay1s = pdMS_TO_TICKS(500);
vTaskDelayUntil(xLastWakeTime,xDelay1s);

#include "Adafruit_GFX.h" //For the touchscreen
#include "MCUFRIEND_kbv.h"
#include <TouchScreen.h>
MCUFRIEND_kbv tft; //Creates object screen
tft.setCursor(x,y); //sets x and y points for screen to draw or write
tft.println("String"); //prints String contents on screen
tft.reset(); //resets content of screen
tft.begin(ID); //begins screen
tft.fillScreen(COLOR); //fills screen with color
tft.setTextSize(NUMBER); //sets text size
tft.setTextColor(COLOR); //sets text color
tft.fillRect(x,y,x_size,y_size,COLOR); //Draws rectangle that starts at (x,y)
with a size of x_size on the x-axis and y_size on the y-axis

#include "SoftwareSerial.h" //For communicating with the Mp3 through pins
#include "DFRobotDFPlayerMini.h" //MP3 Library
SoftwareSerial softwareSerial(PIN, PIN); //begins serial connection with two
pins
DFRobotDFPlayerMini player; //creates an MP3 player object
player.start(); //plays music for the first time or after pausing
player.next(); //plays next song
player. Previous(); // plays previous song
player.pause(); //pauses song
```

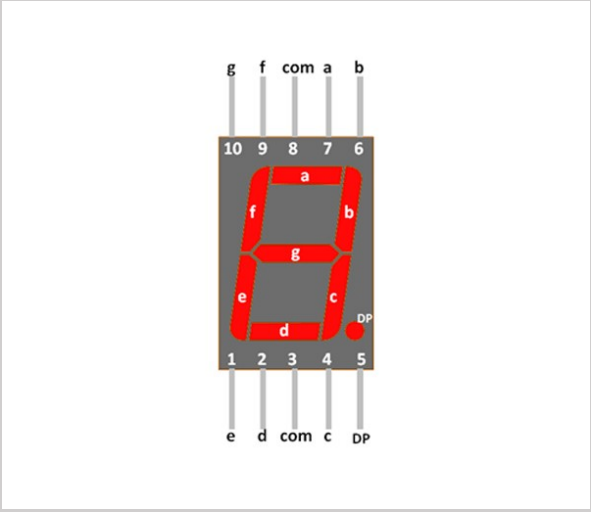
Input and Output Handling

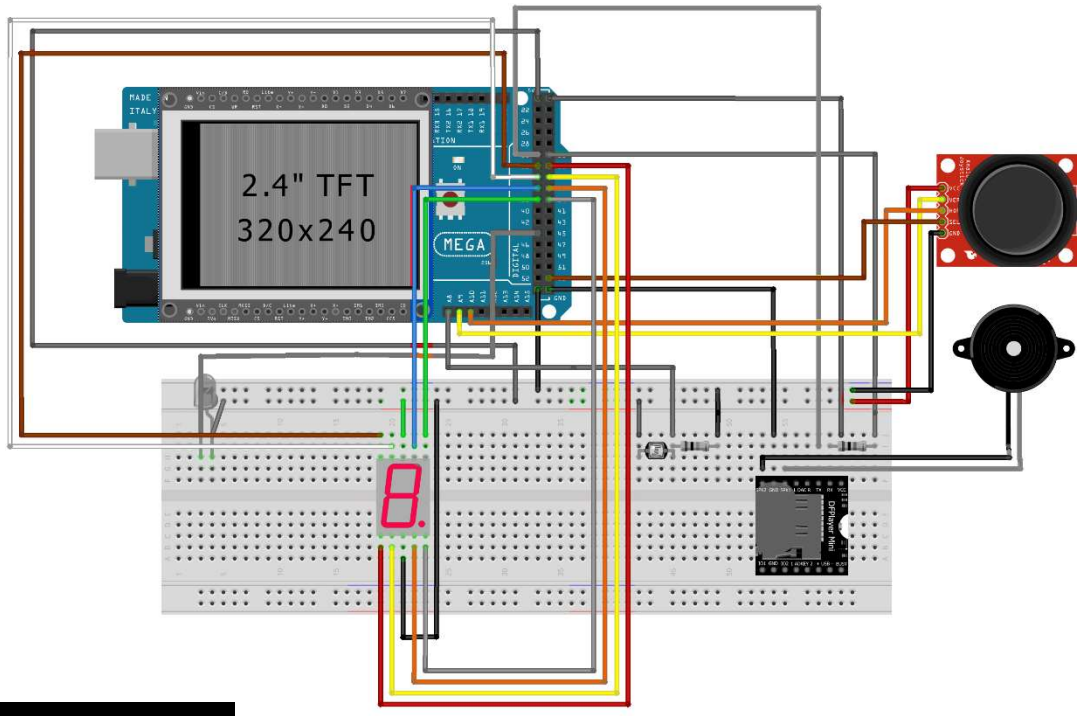
Car Function:

```
const int An = 2; //Motor 1 Reverse
const int Ap = 3; //Motor 1 Forward
const int Bn = 4; //Motor 2 Reverse
const int Bp = 5; //Motor 1 Forward
const int R = 6; //Right
const int C = 7; //Center
const int L = 8; //Left
const int Alert = 13; //LED
void car(){
    pinMode(2,OUTPUT);pinMode(3,OUTPUT);pinMode(5,OUTPUT);pinMode(4,OUTPUT);
    pinMode(6,INPUT); pinMode(7,INPUT); pinMode(8,INPUT);
}
```

Joystick Function:

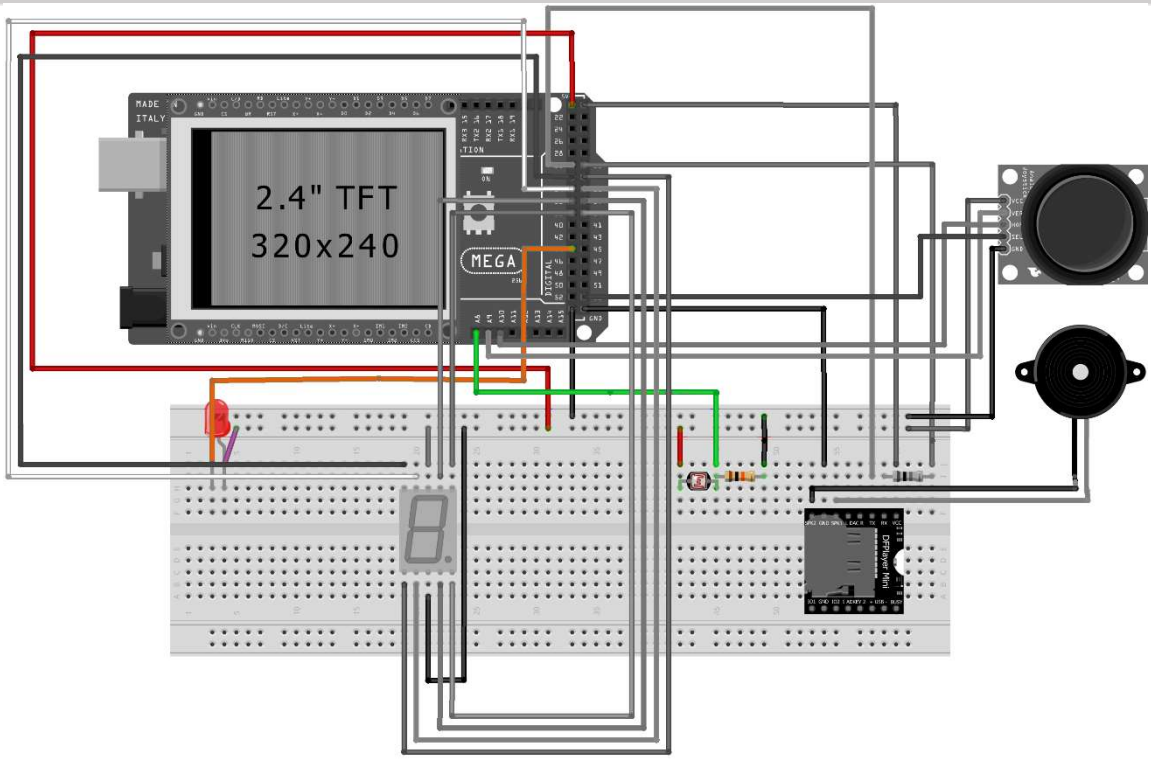
```
int VRx = A9; //Joystick X
int VRy = A10; //Joystick Y
int SW = 53; // Joystick Button
const int g = 32;
const int f = 34;
const int a = 36;
const int b = 38;
const int e = 33;
const int d = 35;
const int c = 37;
const int dp = 49;
pinMode(VRx, INPUT); pinMode(VRy, INPUT); pinMode(SW, INPUT_PULLUP);
//everything else is set to OUTPUT
```





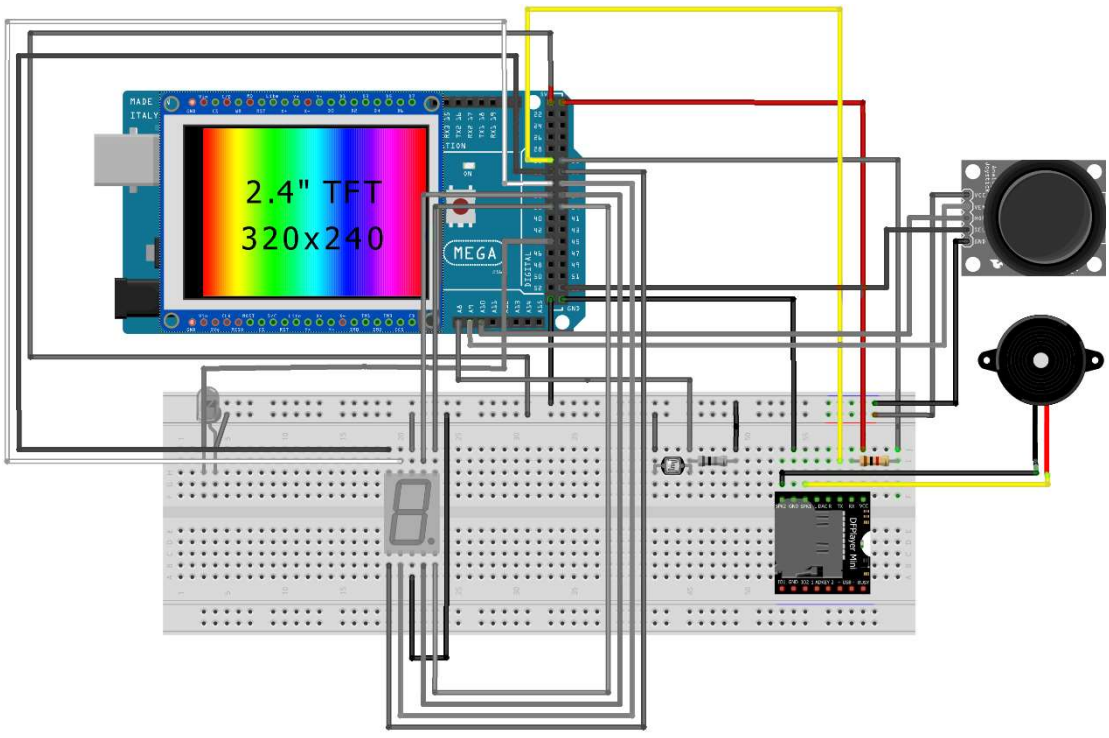
LDR Function:

```
const int ledPin = 44; //Digital Output (PWM)
const int ldrPin = A8; //Analog Input
void lights(){
 pinMode(ledPin, OUTPUT);
 pinMode(ldrPin, INPUT);
}
```



Touch Screen and MP3 Player:

```
const int XP=8,XM=A2,YP=A3,YM=9; //240x320 ID=0x7789

const int TS_LEFT=120,TS_RT=905,TS_TOP=90,TS_BOT=895;


TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

TSPoint tp;

#define YP A2

#define XM A1

#define YM 6

#define XP 7

#define LCD_CS A3

#define LCD_CD A2

#define LCD_WR A1

#define LCD_RD A0


SoftwareSerial softwareSerial(30, 31);

//In this case the screen is both an input and an output. The screen's pins are configured as written above. They work thanks to the imported library. The latter statement is also true of the MP3 Player.
```



RTOS Configuration:

For the Arduino Uno, the task of scheduling the task was easy. There was only one task.

```
void setup() {
  xTaskCreate(car,"car",256,NULL,2,NULL);
}
```

For the Arduino Mega, things were a bit more complicated. Since both the MP3 Player and the screen need a semi “constant” feed of information, the intuition of making the MP3 player have the smallest priority went out the window. We first attempted to make the MP3 player have the highest priority but to no avail as it ate up all processing time and caused starvation of the other tasks. In a subsequent attempts, we put various delay signals. They were either too small to give the other tasks any perceptible difference or large enough to cause visible delays to the screen (such as very slow refreshes or very slow response to the touch). Even after an increase in stack size for the MP3 Player it was still not enough. In the end, we just settled for giving all the tasks the same priority and letting the Arduino figure it out Round Robin Style (¯_(ツ)_/¯)

```
void setup() {
  xTaskCreate(lights,"LED",256,NULL,2,NULL);
  xTaskCreate(mp3_player,"MP3",5120,NULL,2,NULL);
  xTaskCreate(joysegment,"joystick",256,NULL,2,NULL);
}
```

Challenges Faced:

The only major challenge aside from the scheduling of the Arduino Mega, was that we were unable to find two IR Sensors so we had to make do with a 3 Channel Line Tracker, It is the same concept but instead dictating movement based on two lines in the extreme ends of the car it dictates movement based on one line in the center. We were also unable to find a way to read the names of the songs through the MP3 Player Module so the song names displayed on the screen are hard coded. The PWM with the Car didn’t quite work, so we just made it a digital output. Also, I burned 2 different 7 segment displays.