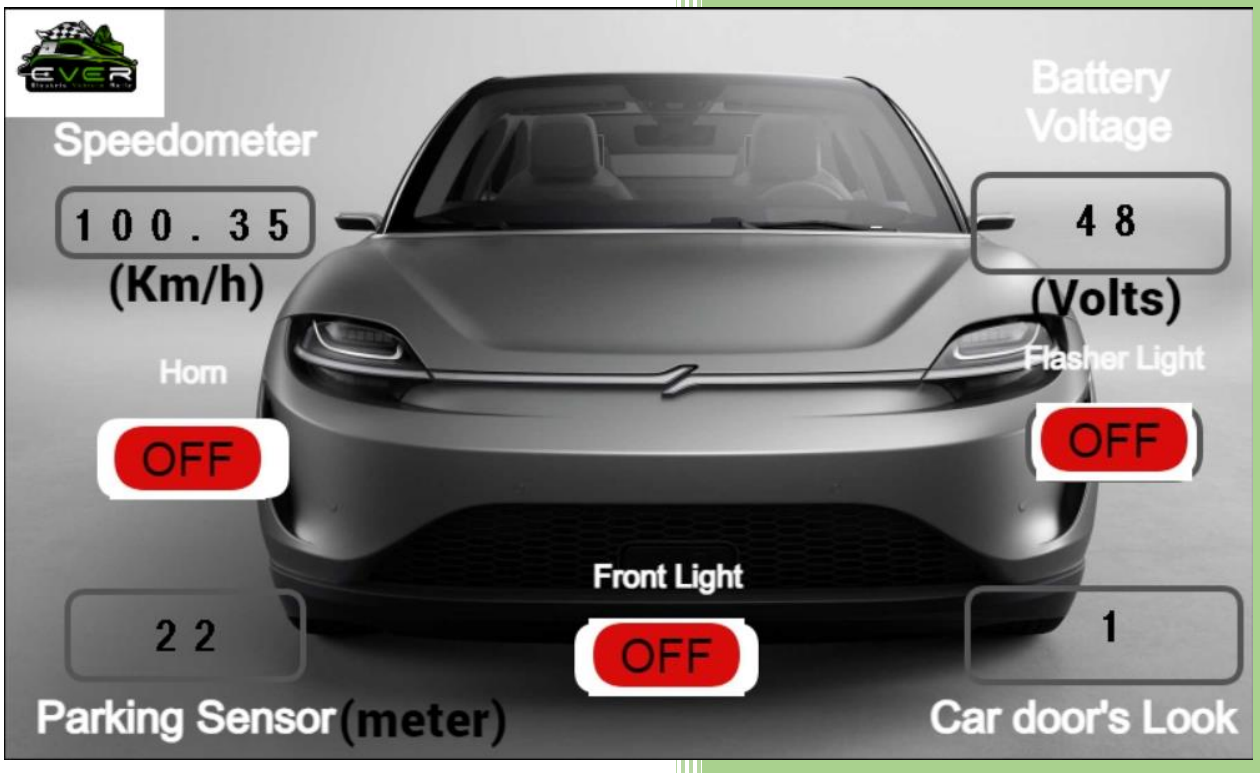
2023

MAJOR TASK TEAM (2)



Team Members

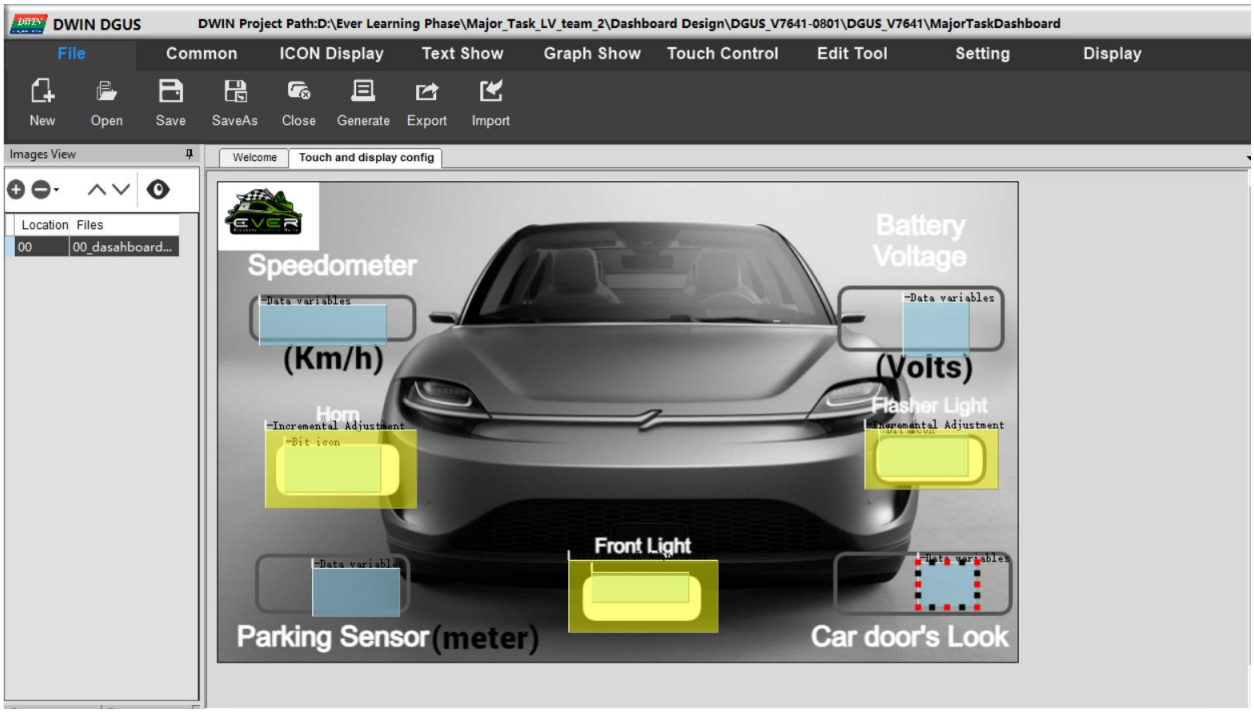
- Ahmed Mostafa Ahmed Mostafa 1900309
- Donia Hany Saeed 2000400
- Aly Mohamed Abdelaziz 2001358

Team 2

EVER ASU RACING TEAM

3/11/2023

EVER Dashboard



We create four fields:

1- Speedometer: That shows the speed of car in Km/h.

2- Battery Voltage: That shows the voltage of electric car battery.

3- Parking Sensor: That measure distance in meters between the car and the obstacle for parking.

4- Car door's look: That inform us about the state of car doors are opened (1) or closed (0).

We also create three buttons for controlling the horn, the front lights of cars.

Note:

Speedometer and Battery voltage are float variables, but parking sensor and car door's lock are integer.

The three buttons each of them is considered as bit icon for small space in memory.

The Addresses of each field in UI of Dashboard:

1- Speedometer: 0x1000

2- Battery Voltage: 0x1100

3- Parking Sensor: 0x1200

4- Car door's look: 0x1300

5- Horn: 0x0600

6- Flasher Light: 0x0400

7- Front Light: 0x0200

Note: These addresses are used in order to access the LCD with Arduino or any microcontroller.

Note: All photos and icons are in DWIN_SET.

Main Arduino Code

Libraries Used:

```
#include <SPI.h>
#include <LiquidCrystal.h>
```

Definitions:

```
#define SLAVE_SELECT  53
#define MISO          50

#define CAR_LOCK      4
/* LCD Pins */
#define RS_PIN   12
#define EN_PIN   11
#define DS4      7
#define DS5      6
#define DS6      3
#define DS7      2

#define LCD_COLUMNS  16
#define LCD_ROWS     2

#define FRONT_LIGHT   9
#define HORN          10
```

Global Variables:

```
float voltageSensor = 0.0;
float speedSensor = 0.0;
int distance = 0.0;
byte Buffer[20];
byte Buffer_len = 0;
unsigned char flag = false;
char horn;
char flasher;
char frontLight;
char carLock;
```

Class Instance of Character LCD:

```
LiquidCrystal lcd(RS_PIN, EN_PIN, DS4, DS5, DS6, DS7);
```

SPI for connecting Main Arduino and Secondary Arduino:

```
SPI.begin();
pinMode(SLAVE_SELECT, INPUT);
SPCR|= _BV(SPE);  // SPI in slave mode
SPI.attachInterrupt(); // Enable Interrupt
```

UART Configuration for communicating Main Arduino with DWIN LCD:

```
Serial.begin(9600);
```

Character LCD Configuration:

```
lcd.begin(LCD_COLUMNS, LCD_ROWS);
lcd.setCursor(0, 0);
```

Configure Pins Direction:

```
/* Car lock pin */
pinMode(CAR_LOCK, INPUT);
/* Front Light */
pinMode(FRONT_LIGHT, OUTPUT);
/* Horn */
pinMode(HORN, OUTPUT);
```

Void Loop

Receive Distance, Battery Voltage and Car Speed from 2nd Arduino:

```
SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE0));
SPI.transfer((byte*)&distance, sizeof(distance));
SPI.transfer((byte*)&voltageSensor, sizeof(voltageSensor));
SPI.transfer((byte*)&speedSensor, sizeof(speedSensor));
SPI.endTransaction(); delay(100);
```

Send Distance, Battery Voltage and Car Speed to DWIN LCD:

```
sendIntNumber(distance, 0x1200);  // Send distance value to lcd
sendFloatNumber(speedSensor, 0x1000); // send speed in Km/h to lcd
sendFloatNumber(voltageSensor, 0x1100); // send voltage of battery to lcd

/* Send state of car lock to lcd */
carLock = digitalRead(CAR_LOCK);
sendIntNumber(carLock, 0x1300);
```

Receive Horn, Flasher and Front Lights from DWIN LCD:

```
DISPLAY_switchRead(); // receive from display horn, flasher, front light states
digitalWrite(HORN, horn);
if(flasher == 1 && frontLight == 0){
digitalWrite(FRONT_LIGHT, flasher);
}
else if(flasher == 0 && frontLight == 1){
digitalWrite(FRONT_LIGHT, frontLight);
}
```

Create Function that receive values from DWIN LCD:

```
void DISPLAY_switchRead(void){
if(Serial.available()){
  Buffer[Buffer_len] = Serial.read();
Buffer_len++;
flag = true;
}
else{
if(flag){
  if(Buffer[0] == 0x5A){
    if(Buffer[4] == 0x06){
      horn = bitRead(Buffer[8], 0);
    }
  else if(Buffer[4] == 0x04){
       flasher = bitRead(Buffer[8], 0);
  }
  else if(Buffer[4] == 0x02){
       frontLight = bitRead(Buffer[8], 0);
  }
  }
Buffer_len = 0;
flag = false;
}}
}
```

Create Function that sends Integer Data to DWIN LCD:

```
void sendIntNumber(int intValue, long int address){
  Serial.write(0x5A); // Header
  Serial.write(0xA5); // Header
  Serial.write(0x05); // Length: VP address + write command + length of float
  Serial.write(0x82); // write command
  Serial.write(highByte(address)); // Address High byte
  Serial.write(lowByte(address));  // Address Low byte
  Serial.write(highByte(intValue)); // data High byte
  Serial.write(lowByte(intValue));  // data low byte
}
```

Create Function that sends Float Data to DWIN LCD:

```
void sendFloatNumber(float floatValue, long int address){
  Serial.write(0x5A); // Header
  Serial.write(0xA5); // Header
  Serial.write(0x07); // Length: VP address + write command + length of float
  Serial.write(0x82); // write command
  Serial.write(highByte(address));
  Serial.write(lowByte(address));
byte hex[4] = {0};
FloatToHex(floatValue, hex);

Serial.write(hex[3]);
Serial.write(hex[2]);
Serial.write(hex[1]);
Serial.write(hex[0]);
}
```

Create Function that Converts Float to Hex value:

```
void FloatToHex(float f, byte* hex){
  byte* f_byte = reinterpret_cast<byte*>(&f); // The value of f_byte is pointer
to f
  memcpy(hex, f_byte, 4); // hex: destination, f_byte: source, 4: number of bytes
to copy
  }
```

Git Hub Repository:

https://github.com/Ahmed-Elnafadi/Major_Task_Part1.git