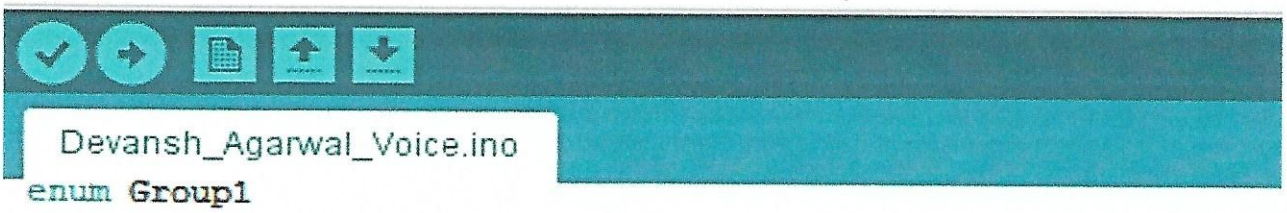
Devansh_Agarwal_Voice.ino

```
//Devansh Agarwal Voice Prompt Code
//Code for voice recognition and auxillary controls of car
#include "Arduino.h"
#if !defined(SERIAL_PORT_MONITOR)
  #error "Arduino version not supported. Please update your IDE to the latest version."
#endif

#if defined(SERIAL_PORT_USBVIRTUAL)
  // Shield Jumper on HW (for UNO)
  #define port SERIAL_PORT_HARDWARE
  #define pcSerial SERIAL_PORT_USBVIRTUAL
#else
  // Shield Jumper on SW (using pins 12/13 or 8/9 as RX/TX)
  #include "SoftwareSerial.h"
  SoftwareSerial port(12, 13);
  #define pcSerial SERIAL_PORT_MONITOR
#endif

#include "EasyVR.h" //compiled library for voice recognition

EasyVR easyvr(port);
//Groups and Commands
enum Groups
{
  GROUP_0  = 0,
  GROUP_1  = 1,
};

enum Group0
{
  G0_AERODRIVE = 0,
};

enum Group1
{
```

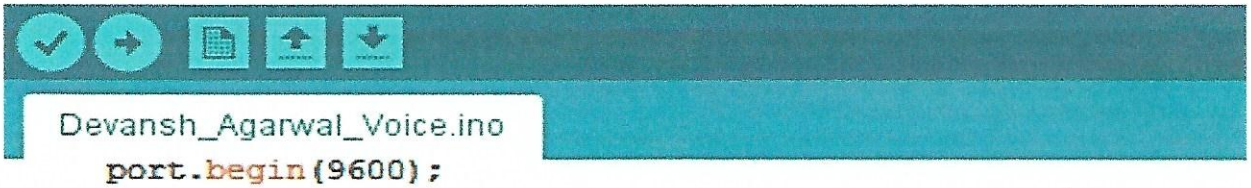
Done compiling.



Devansh_Agarwal_Voice.ino

```
enum Group1
{
  G1_IGNITION = 0,
  G1_START = 1,
  G1_STOP = 2,
  G1_HEADLIGHT_ON = 3,
  G1_HEADLIGHT_OFF = 4,
  G1_WIPER_ON = 5,
  G1_WIPER_OFF = 6,
  G1_RIGHT_TURN = 7,
  G1_LEFT_TURN = 8,
};

int8_t group, idx;

void setup()
{

  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);

  // setup PC serial port
  pcSerial.begin(9600);

  // bridge mode
  int mode = easyvr.bridgeRequested(pcSerial);
  switch (mode)
  {
  case EasyVR::BRIDGE_NONE:
    // setup EasyVoiceRecognition serial port
    port.begin(9600);
```

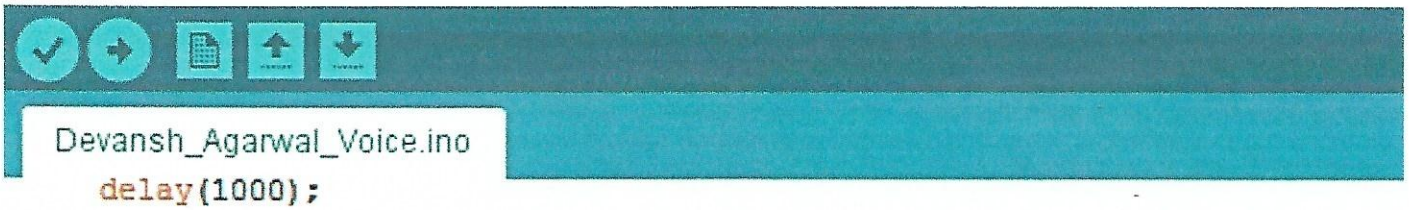
Done compiling.



Devansh_Agarwal_Voice.ino

```
    port.begin(9600);
    // normal run
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge not started!"));
    delay(250);
      easyvr.playSound(1, EasyVR::VOL_FULL);
      delay(500);
    break;

  case EasyVR::BRIDGE_NORMAL:
    // setup EasyVR serial port (low speed)
    port.begin(9600);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;

  case EasyVR::BRIDGE_BOOT:
    // setup EasyVR serial port (high speed)
    port.begin(115200);
    // soft-connect the two serial ports (PC and EasyVR)
    easyvr.bridgeLoop(pcSerial);
    // resume normally if aborted
    pcSerial.println(F("---"));
    pcSerial.println(F("Bridge connection aborted!"));
    break;
  }

  while (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    delay(1000);
  }
```

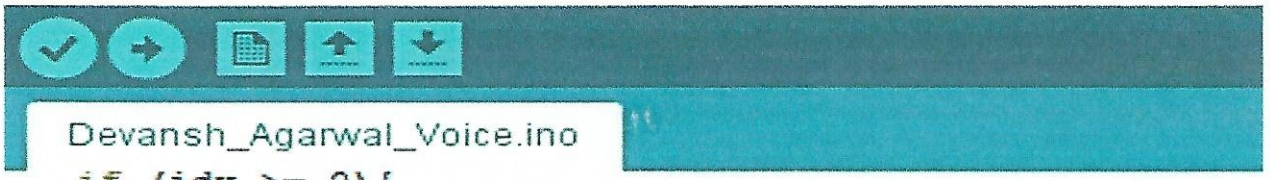
Done compiling.



Devansh_Agarwal_Voice.ino

```
    delay(1000);
  }

  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(5);
  easyvr.setLanguage(0);
 group=(0);
}
void action();

void loop()
{

  easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)

  Serial.print("Say a command in Group ");
  Serial.println(group);
  easyvr.recognizeCommand(group);

  do
  {
    // background processing as the controller waits for a voice command
  }
  while (!easyvr.hasFinished());

  easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

  idx = easyvr.getWord();
  if (idx >=0){

    // built-in trigger (ROBOT)
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0){
```

Done compiling.



Devansh_Agarwal_Voice.ino

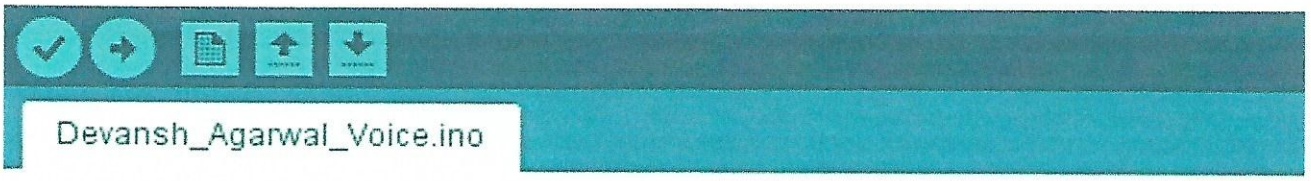
```
  if (idx >= 0){
    // print debug message
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
      Serial.println(name);
    }
    else
      Serial.println();
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // perform some action
    action();
  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
      group=(0);
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}

void action()
{
    switch (group)
    {
    case GROUP_0:
```

Done compiling.

Devansh_Agarwal_Voice.ino

```
{
case GROUP_0:
  switch (idx)
  {
  case G0_AERODRIVE: //define voice commands

  delay(20);
  easyvr.playSound(16, EasyVR::VOL_FULL); //initiate reply by microcontroller
  group=(1);
    break;
  }
  break;
case GROUP_1:
  switch (idx)
  {
  case G1_IGNITION: //initiates diagnostics and starts car

  digitalWrite(3,HIGH);
  delay(10);
  group=(0);
   break;

  case G1_START: //powers the engine

  digitalWrite(4,HIGH);
  delay(100);
  digitalWrite(2,HIGH);
  delay(1100);
  digitalWrite(2,LOW);
  delay(50);
  digitalWrite(4,LOW);
  group=(0);
   break;

  case G1_STOP: //powers off the engine
```

Done compiling.



Devansh_Agarwal_Voice.ino

```
case G1_STOP: //powers off the engine

delay(250);
easyvr.playSound(13, EasyVR::VOL_FULL);
delay(500);
digitalWrite(3,LOW);
delay(10);
group=(0);
  break;

case G1_HEADLIGHT_ON: //switches off rear and front lamps

digitalWrite(6,HIGH);
delay(100);
digitalWrite(6,LOW);
group=(0);
  break;

case G1_HEADLIGHT_OFF: //switches on rear and front lamps

digitalWrite(6,HIGH);
delay(100);
digitalWrite(6,LOW);
delay(10);
group=(0);
  break;

case G1_WIPER_ON: //turns on wiper

digitalWrite(7,HIGH);
delay(100);
digitalWrite(7,LOW);
delay(10);
group=(0);
   break;
```

Done compiling.

Devansh_Agarwal_Voice.ino

```
        delay(10);
        group=(0);
          break;

        case G1_WIPER_ON: //turns on wiper

        digitalWrite(7,HIGH);
        delay(100);
        digitalWrite(7,LOW);
        delay(10);
        group=(0);
           break;

        case G1_WIPER_OFF: //turns off wiper

        digitalWrite(7,HIGH);
        delay(100);
        digitalWrite(7,LOW);
        delay(10);
        group=(0);
           break;

        case G1_RIGHT_TURN: //turns right indicator on
        delay (10);
        group=(0);
           break;

        case G1_LEFT_TURN: //turns left indicator on

        delay(10);
        group=(0);
           break;
        }
        break;
    }
}
```

Done compiling.