S

Embedded Systems Project

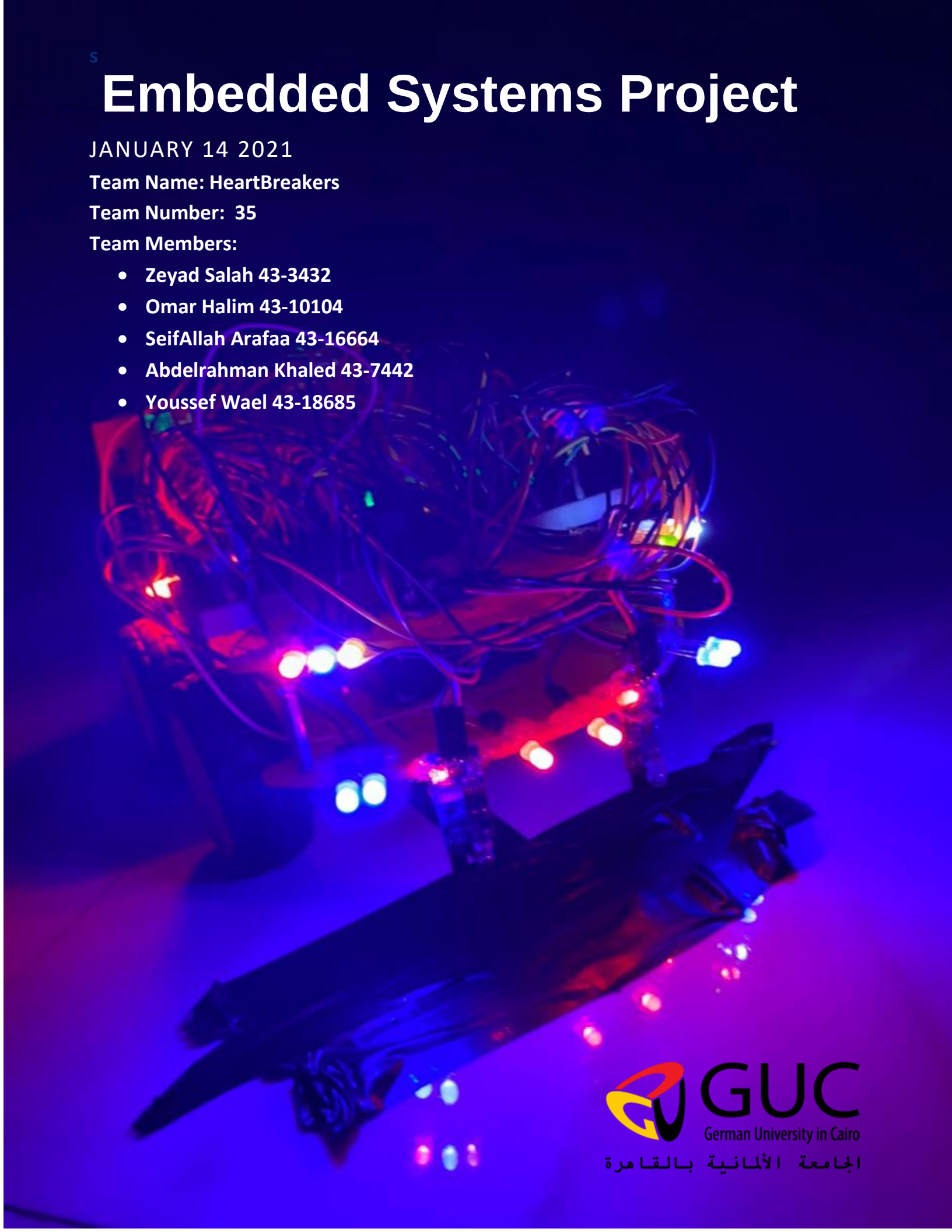
JANUARY 14 2021

Team Name: HeartBreakers

Team Number: 35

Team Members:

- **Zeyad Salah 43-3432**
- **Omar Halim 43-10104**
- **SeifAllah Arafaa 43-16664**
- **Abdelrahman Khaled 43-7442**
- **Youssef Wael 43-18685**



GUC
German University in Cairo
الجامعة الألمانية بالقاهرة

Introduction

In this project, we implemented some features using Arduino Uno and Mega boards, giving each feature a suitable priority. We used Arduino C alongside FreeRTOS to implement our project.

Let me just say how amusing the project was. It showed us how real hard systems work and simultaneously. Although it really gave us some hard times and impossible errors but we kept running towards the right answer and eventually it was all worth it. The main idea of the project is a vehicle with some extraordinary features. We were asked to choose these features from a list. These are the features that we choose:

1. Lane Keeping Assist (LKA): autonomous lane departure system that detects if the vehicle is leaving its lane using sensors, it will gently steer the car back into the lane if it begins to drift out of it while alerting the driver.
2. Fears on the Road! display the current gear on a 7-Segment display (P, D, N, R) and adaptive headlights according to the surrounding light intensity.

3. Ears on the Road! play MP3 files via a speaker on the car and control it using an “LCD TFT 2.4 With Touch Screen” with the following operations:
    - Play/Pause
    - Previous
    - Next
    - Display the song number/name

Components:

- Arduino Board – Arduino UNO Rev3 – China Clone (able to read inputs from sensors and turns them into outputs like motor movement and lights)
- Arduino Board – Arduino Mega2560 Rev3 – China Clone LED Lights 5mm (able to read inputs from sensors and

turns them into outputs like motor movement and lights but on a larger scale than the Uno)

- F-M & M-M & F-F Jumpers (To connect the boards and components together)
- Bread Board (used for creating electrical connections between electronic components and single board computers or microcontrollers)
- 4pin Photoresist or, LDR Module – Optical Sensitive Resistance Light Detection Sensor Module (used to detect the intensity of light.)
- 1 Pack of Mixed Color LED Size 5mm (60 LED) + Free Storage Box (To display lights)
- Joystick 2Axis (Analog)-Thumb with PCB & Pin Header (acts as a shifter for the car)
- 7 Segment (1 × 1) 0.56 Common Anode (To display gear positions ex: d, p, n and r)
- 450 Carbon Resistor 5% Tolerance 1/4W – For All Projects in One KIT (to lower the voltage so the components are not fried)

- MP3 Mini Player for Arduino or Microcontrollers (To play songs on it using a small SD card)
- Arduino Shield "LCD TFT 2.4 With Touch Screen" (To Control the music on the MP3 player)
- Very Flat Small Speaker, 8 Ohm, 0.25W – 20mm (To output the sound from the MP3 player)
- PVC Electrical Insulation Tape – Black (To create a lane on the floor so the car can follow)
- IR Infrared Obstacle Avoidance (Object Detection) Sensor (to detect the lane)
- DC Motors (To move the car)
- H-bridge Arduino L293D (to connect the motors to the Arduino)
- Car/Vehicle Kit (to put the components on it)

Team Members Parts:

- Zeyad Salah (Lane Detection)
- Omar Halim (7-Segment Display)
- SeifAllah Arafaa (LCD Screen & MP3 DF-Miniplayer)
- Abdelrahman Khaled (Light Intensity Levels & LED Lights)
- Youssef Wael (Priorities using FreeRTOS)

Implementation

We used 2 Arduinos, an Arduino mega and an Uno. The Uno was used for the touch screen. The mega was used for the rest of the features. We linked them both using master/slave concept. The priorities were implemented in the mega. When we were building the car, we followed this

order. I suggest if we are implementing the same project or something similar to follow the same pattern.

1. Lane Detection with visual alerts when the car tries to leave
2. Headlights with levels of intensity according to the light in the room
3. Touch screen and how to take an input from it
4. Output a sound using the speaker and the MP3
5. Link step 4 and 5 together by unifying their codes
6. Link step 1 and 2 together by unifying their codes
7. Link step 5 and 6 together by unifying their codes and check that all systems are working all at once
8. Implement priorities using FreeRTOS library

As you know we needed to connect the both Arduinos together so we used the master/slave concept. The Arduino

mega was the master and the Uno was the slave. The Arduino sent input signals to the mega.

There were different types of inputs. The motor input was analog and the rest of the inputs were digital. The motor's speed was controlled by us while the rest of the sensors were inputting either 0 or 1 or in other words high or low. The outputs are the movement of the car(analog), the display of lights according to the light intensity in the room(digital), the display of gear(digital) and speaker sound from the MP3 player.

The priorities are in the Arduino Mega code. They are divided into tasks, 2 tasks to be exact. The 1st task contains the motor, MP3 and the touch screen. The 2nd task contains the light intensity levels and joystick which shows the gears with a delay of 3000ms which is equal to 3 seconds.

We implemented an extra feature which is dark mode driving. There are 6 headlights in the front as shown in the picture. These lights are always on so the car can see the lane below to be able to drive when there is no light in the room.

Arduino Mega Pins Input/Output:

1. Motor Pins

1.1. Name = left (Left Lane Sensor), Pin = A15, Type = analog input

1.2. Name = right (Right Lane Sensor), Pin = A14, Type analog input

1.3. Name = led1, Pin = 37, Type = digital output

1.4. Name = led2, Pin = 31, Type = digital output

1.5. Name = led3, Pin = 41, Type = digital output

1.6. Name = MotorX.run(Direction), Type = analog output, X = Motor Number, Direction = FORWARD, BACKWARD, REALEASE

1.7. Name = MotorX.setSpeed(Y), Type = analog output, X = Motor Number, Y = Speed Value

2. Light Sensor Pins

2.1. Name = lightSensorPin, Pin = A8, Type = analog input

2.2. Name = LedPin1, Pin = 51, Type = digital output

2.3. Name = LedPin2, Pin = 50, Type = digital output

2.4. Name = LedPin3, Pin = 49, Type = digital output

2.5. Name = LedPin4, Pin = 48, Type = digital output

2.6. Name = LedPin5, Pin = 47, Type = digital output

2.7. Name = LedPin6, Pin = 46, Type = digital output

3. Gear/Joystick Pins

3.1. Name = VRx (Gear Input Direction), Pin = A13, Type = analog input

3.2. Name = VRy (Gear Input Direction), Pin = A12, Type = analog input

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

3.4. Name = 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

Problems and Limitations:

- The polarity of the motors, which wire is the positive and which id the negative
- Connections of the 7-segment display
- The Sensitivity of the light sensor in the readings
- Touch screen input to the Arduino Uno
- Touch Screen connection with the MP3 player
- Unifying the codes
- Prioritize the tasks according to their importance

Arduino Mega Code Libraries:

- #include <AFMotor.h> (provides speed and direction control for up to four DC motors)
- #include <SoftwareSerial.h> (enables serial communication with a digital pin other than the serial port)
- #include <Wire.h> (For Connections between the two Arduinos as master/slave)

- #include <Arduino_FreeRTOS> (To Divide tasks and prioritize them among their importance)

Arduino Uno Code Libraries:

- #include <SPFD5408_Adafruit_GFX.h> (to avoid problems with duplicate libraries and enables also have the original library Adafruit ready for use in other projects with another TFT hardware.)
- #include <SPFD5408_Adafruit_TFTLCD.h> (to avoid problems with duplicate libraries and enables also have the original library Adafruit ready for use in other projects with another TFT hardware.)
- #include <SPFD5408_TouchScreen.h> (To enable the touch screen to work and to take inputs from it)
- #include<Wire.h> (For Connection of the Arduino Uno and Mega Together)

Arduino Mega Code (Master):

```
//including the libraries
#include <AFMotor.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Arduino_FreeRTOS.h>
```

```
# define Start_Byte     0x7E
# define Version_Byte   0xFF
# define Command_Length 0x06
# define End_Byte       0xEF
# define Acknowledge    0x00
#define left A15
#define right A14

int xPosition = 0;
int yPosition = 0;
int SW_state = 0;
int mapX = 0;
int mapY = 0;
int VRx = A13;
int VRy = A12;
int SW = 53;

int d1 = 5000;
int d2 = 10000;
int d3 = 20000;
SoftwareSerial mySerial(35, 33);
int btnNext;
int btnPause;
int btnPrevious;
int volume = 15;
boolean Playing = false;
char ko;
int previousv=-1;


int lightSensorPin = A8;
int analogValue = 0;
```

```
const int a = 24;
const int b = 22;
const int c = 16;
const int d = 15;
const int e = 14;
const int f = 19;
const int g = 18;
const int h = 17;

int LedPin1 = 51;
int LedPin2 = 50;
int LedPin3 = 49;
int LedPin4 = 48;
int LedPin5 = 47;
int LedPin6 = 46;

int led1=37;
 int led2=31;
 int led3=41;

//defining motors
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor3(3, MOTOR34_1KHZ);
AF_DCMotor motor4(4, MOTOR34_1KHZ);


void setup() {

  //declaring pin types
  pinMode(left,INPUT);
  pinMode(right,INPUT);
  pinMode(lightSensorPin, INPUT);
  pinMode(VRx, INPUT);
```

```
 pinMode(VRy, INPUT);
 pinMode(SW, INPUT_PULLUP);

 pinMode(a, OUTPUT);  //A
 pinMode(b, OUTPUT);  //B
 pinMode(c, OUTPUT);  //C
 pinMode(d, OUTPUT);  //D
 pinMode(e, OUTPUT);  //E
 pinMode(f, OUTPUT);  //F
 pinMode(g, OUTPUT); //G

 pinMode(LedPin1, OUTPUT);
 pinMode(LedPin2,OUTPUT);
 pinMode(LedPin3,OUTPUT);
 pinMode(LedPin4, OUTPUT);
 pinMode(LedPin5,OUTPUT);
 pinMode(LedPin6,OUTPUT);

 pinMode(led1,OUTPUT);
 pinMode(led2,OUTPUT);
 pinMode(led3,OUTPUT);

digitalWrite(led1, LOW);
digitalWrite(led2, LOW);
digitalWrite(led3, LOW);

 Wire.begin() ;
 btnNext=HIGH;
 btnPause=HIGH;
 btnPrevious=HIGH;
 Serial.begin(9600);
 mySerial.begin(9600);
 //delay(1000);
```

```
  xTaskCreate(MyTask1, "motor motion , MP3 sound and Screen display", 1000,
NULL, 2, NULL);
  xTaskCreate(MyTask2, "Light and Gear", 1000, NULL, 2, NULL);
   vTaskStartScheduler();

}

void loop() {

     //Serial.println("Task0");
  //delay(500);



}
////////////////////////////////Task
1////////////////////////////////////////////////////////////////////////////////
////////////////////

static void MyTask1(void* pvParameters){

  playFirst();
 Playing = true;
   while(1){
     // Serial.println("Task1");

    //line detected by both
  if(digitalRead(left)==0 && digitalRead(right)==0){
    //Forward

  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);

    motor1.run(FORWARD);
```

```
  motor1.setSpeed(150);
  motor2.run(FORWARD);
  motor2.setSpeed(150);
  motor3.run(FORWARD);
  motor3.setSpeed(150);
  motor4.run(FORWARD);
  motor4.setSpeed(150);
 }
 //line detected by left sensor
 else if(digitalRead(left)==0 && !analogRead(right)==0){
  //turn left
digitalWrite(led1, HIGH);
digitalWrite(led2, HIGH);
digitalWrite(led3, HIGH);

  motor1.run(FORWARD);
  motor1.setSpeed(200);
  motor2.run(FORWARD);
  motor2.setSpeed(200);
  motor3.run(BACKWARD);
  motor3.setSpeed(200);
  motor4.run(BACKWARD);
  motor4.setSpeed(200);


 }
 //line detected by right sensor
 else if(!digitalRead(left)==0 && digitalRead(right)==0){
  //turn right

digitalWrite(led1, HIGH);
digitalWrite(led2, HIGH);
digitalWrite(led3, HIGH);
```

```
  motor1.run(BACKWARD);
  motor1.setSpeed(200);
  motor2.run(BACKWARD);
  motor2.setSpeed(200);
  motor3.run(FORWARD);
  motor3.setSpeed(200);
  motor4.run(FORWARD);
  motor4.setSpeed(200);


 }
 //line detected by none
 else if(!digitalRead(left)==0 && !digitalRead(right)==0){
   //stop
digitalWrite(led1, HIGH);
digitalWrite(led2, HIGH);
digitalWrite(led3, HIGH);

  motor1.run(RELEASE);
  motor1.setSpeed(0);
  motor2.run(RELEASE);
  motor2.setSpeed(0);
  motor3.run(RELEASE);
  motor3.setSpeed(0);
  motor4.run(RELEASE);
  motor4.setSpeed(0);


 }
   btnNext=HIGH;
 btnPause=HIGH;
 btnPrevious=HIGH;

 Wire.requestFrom(5,1);
```

```
 while (Wire.available()) {
 if(Wire.available()!=0){
 ko = Wire.read();
 Serial.print(ko);
 }
 }

 if(ko==2&&ko!=previousv){
 Serial.print(ko);
 btnNext=HIGH;
 btnPause=LOW;
 btnPrevious=HIGH;
 delay(500);
 }
 if(ko==3&&ko!=previousv){
 Serial.println(ko);
 btnNext=LOW;
 btnPause=HIGH;
 btnPrevious=HIGH;
 delay(500);
 }
 if(ko==4&&ko!=previousv){
 Serial.print(ko);
 btnNext=HIGH;
 btnPause=HIGH;
 btnPrevious=LOW;
 delay(500);
 }
previousv=ko;

if (btnPause == LOW)
{
  if(Playing)
  {
```

```
    pause();
     Playing = false;
   }
   else
   {
    Playing = true;
    play();
 }
 }
 if (btnNext == LOW)
  {
   if(Playing)
   {
    next();
   }
  }
  if (btnPrevious == LOW)
 {
   if(Playing)
   {
    previous();
   }
  }
     // vTaskDelay(5000/portTICK_PERIOD_MS);

}

}
////////////////////////////////Task
2////////////////////////////////////////////////////////////////////////////////////
//////////////////////
static void MyTask2(void* pvParameters){
```

```
  while (1){

  Serial.println("Task2");
  xPosition = analogRead(VRx);
  yPosition = analogRead(VRy);
  SW_state = digitalRead(SW);
  mapX = map(xPosition, 0, 1023, -512, 512);
  mapY = map(yPosition, 0, 1023, -512, 512);
  analogValue = analogRead(lightSensorPin);

//  Serial.print("X: ");
//  Serial.print(mapX);
//  Serial.print(" | Y: ");
//  Serial.print(mapY);
//  Serial.print(" | Button: ");
//  Serial.println(SW_state);
  if(950<analogValue){
    digitalWrite(LedPin1, HIGH);
    digitalWrite(LedPin2, HIGH);
    digitalWrite(LedPin3, HIGH);
    digitalWrite(LedPin4, HIGH);
    digitalWrite(LedPin5, HIGH);
    digitalWrite(LedPin6, HIGH);

  }
  else if(850<=analogValue  && analogValue <= 940){
    digitalWrite(LedPin1, HIGH);
    digitalWrite(LedPin2, HIGH);
    digitalWrite(LedPin3, LOW);
    digitalWrite(LedPin4, HIGH);
    digitalWrite(LedPin5, HIGH);
    digitalWrite(LedPin6, LOW);
  }
  else if(600<=analogValue  && analogValue < 750){
```

```
  digitalWrite(LedPin1, HIGH);
  digitalWrite(LedPin2, LOW);
  digitalWrite(LedPin3, LOW);
  digitalWrite(LedPin4, HIGH);
  digitalWrite(LedPin5, LOW);
  digitalWrite(LedPin6, LOW);
 }
 else {
  digitalWrite(LedPin1, LOW);
  digitalWrite(LedPin2, LOW);
  digitalWrite(LedPin3, LOW);
  digitalWrite(LedPin4, LOW);
  digitalWrite(LedPin5, LOW);
  digitalWrite(LedPin6, LOW);
 }


 if(mapX>=100&&mapX<=800&& SW_state==1){  // UP (d)
   digitalWrite(a,HIGH);
   digitalWrite(b,LOW);
   digitalWrite(c,LOW);
   digitalWrite(d,LOW);
   digitalWrite(e,LOW);
   digitalWrite(f,HIGH);
   digitalWrite(g,LOW);

}else if(mapX>=-800&&mapX<=-100&&SW_state==1){  // back (R)
   digitalWrite(a,HIGH);
   digitalWrite(b,HIGH);
   digitalWrite(c,HIGH);
   digitalWrite(d,HIGH);
   digitalWrite(e,LOW);
   digitalWrite(f,HIGH);
   digitalWrite(g,LOW);
```

```
}else if(SW_state==0){  // Normal (P)
    digitalWrite(a,LOW);
    digitalWrite(b,LOW);
    digitalWrite(c,HIGH);
    digitalWrite(d,HIGH);
    digitalWrite(e,LOW);
    digitalWrite(f,LOW);
    digitalWrite(g,LOW);

}else{ // park n
    digitalWrite(a,HIGH);
    digitalWrite(b,HIGH);
    digitalWrite(c,LOW);
    digitalWrite(d,HIGH);
    digitalWrite(e,LOW);
    digitalWrite(f,HIGH);
    digitalWrite(g,LOW);
}

  vTaskDelay(3000/portTICK_PERIOD_MS);

}

  }


////////////////////////////////Task
3////////////////////////////////////////////////////////////////////////////////////
//////////////////////
//static void MyTask3(void* pvParameters){
//    playFirst();
//  Playing = true;
```

```
//
//   while(1){
//         Serial.println("Task3");
//
//
//
//
//  //vTaskDelay(500/portTICK_PERIOD_MS);
//
//
//
//} }
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////

void playFirst()
{
  exe_cmd(0x3F, 0, 0);
  delay(500);
  exe_cmd(0x06, 0, volume);
  delay(500);
  exe_cmd(0x11,0,1);
  delay(500);
}

void pause()
{
  exe_cmd(0x0E,0,0);
  delay(500);
}

void play()
{
  exe_cmd(0x0D,0,1);
```

```
  delay(500);
}

void next()
{
  exe_cmd(0x01,0,1);
  delay(500);
}

void previous()
{
  exe_cmd(0x02,0,1);
  delay(500);
}


void exe_cmd(byte CMD, byte Par1, byte Par2)
{
  word checksum = -(Version_Byte + Command_Length + CMD + Acknowledge + Par1
+ Par2);
  byte Command_line[10] = { Start_Byte, Version_Byte, Command_Length, CMD,
Acknowledge, Par1, Par2, highByte(checksum), lowByte(checksum), End_Byte};

  for (byte x=0; x<10; x++)
  {
    mySerial.write(Command_line[x]);
  }
}
```

Arduino Uno Code (Slave):

```
// Modified for SPFD5408 Library by Joao Lopes

#include <SPFD5408_Adafruit_GFX.h>  // Core graphics library

#include <SPFD5408_Adafruit_TFTLCD.h> // Hardware-specific library

#include <SPFD5408_TouchScreen.h>

#include<Wire.h>

// Pin assignments for the TFT touch screen

#define YP  A3

#define XM  A2

#define YM  9

#define XP  8

// Calibrated values for the TFT touch screen

#define TS_MINX  104

#define TS_MINY  891

#define TS_MAXX  905

#define TS_MAXY  74

// Instantiate the ts object

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

#define LCD_RD  A0

#define LCD_WR  A1

#define LCD_CD  A2
```

```
#define LCD_CS  A3
#define LCD_RESET A4
// Instantiate the tft object
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
// Assign human-readable names to some common 16-bit color values:
#define BLACK  0x0000
#define BLUE  0x001F
#define RED   0xF800
#define GREEN  0x07E0
#define CYAN  0x07FF
#define MAGENTA 0xF81F
#define YELLOW 0xFFE0
#define WHITE  0xFFFF
#define BOXSIZE   40
#define PENRADIUS  3
#define MINPRESSURE 10
#define MAXPRESSURE 1000
#define LED_PIN   A5
int touch = 0;
int lol=1;
bool play_pause=true;
```

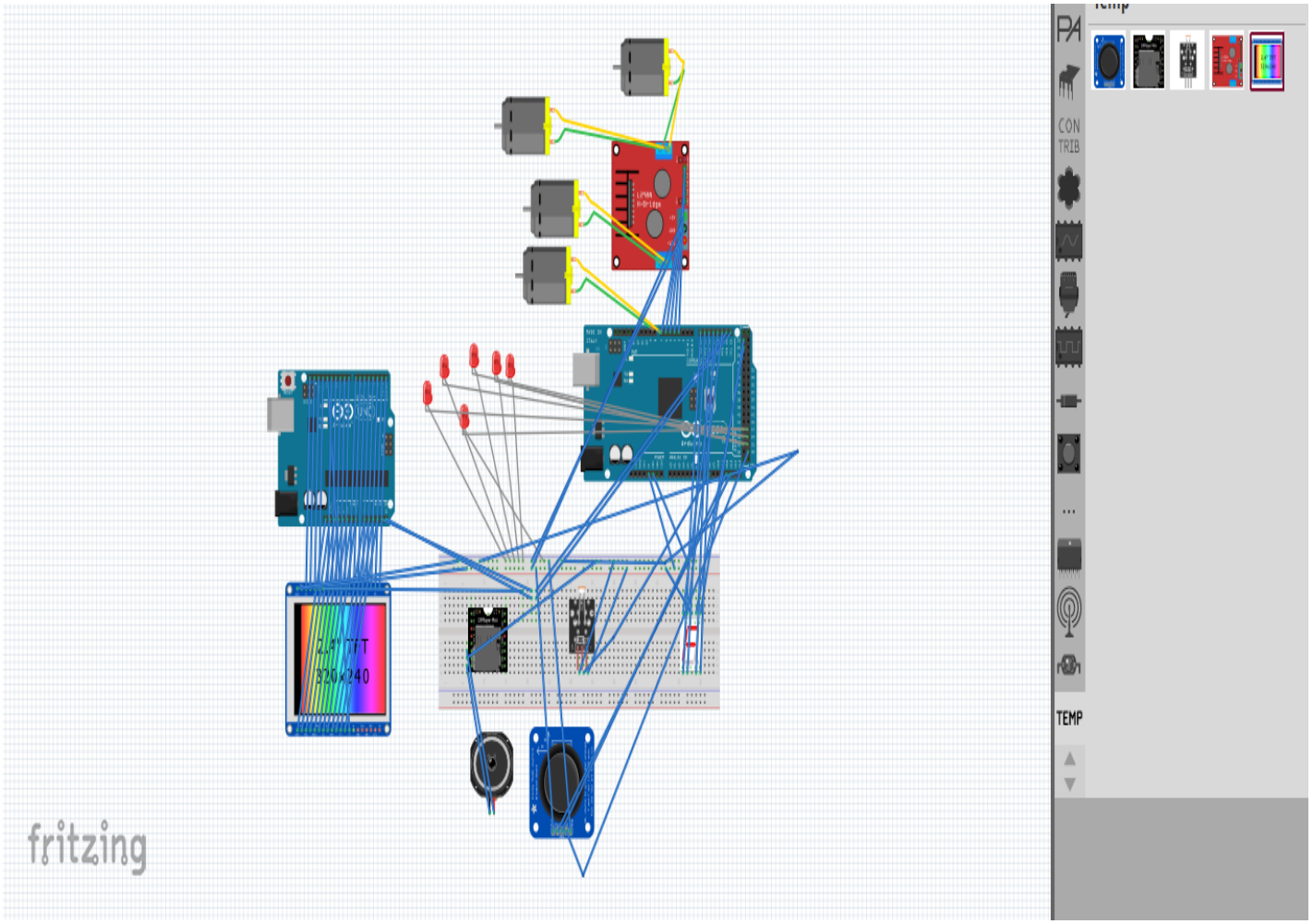
```
void setup() {
 Wire.begin(5) ;
 pinMode(LED_PIN, OUTPUT);
 digitalWrite(LED_PIN, LOW);
 Serial.begin(9600);
 tft.reset();
 tft.begin(0x9341);
 tft.setRotation(2); // This settings works for me with correct orientation
        // tft.setRotation(2);
 tft.setTextColor(WHITE);
 tft.setTextSize(2);
 tft.fillScreen(BLACK);
 tft.fillRect(0, 0, 240, 120, RED);
 //tft.fillRect(120, 0, 120, 120, RED);
 tft.fillRect(0,120,120,120,YELLOW);
 tft.fillRect(120,120,120,120,CYAN);
 tft.setCursor(65, 45);
 tft.println("PLAY/PAUSE");
 tft.setCursor(15, 165);
 tft.println("NEXT");
 tft.setCursor(128, 165);
```

```
 tft.println("PREVIOUS");
 tft.setTextColor(WHITE, BLACK);
 tft.setTextSize(8);
 Wire.onRequest(requestEvent);
}
void loop() {
 // Get the touch points
 TSPoint p = ts.getPoint();
 // Restore the mode
 pinMode(XM, OUTPUT);
 pinMode(YP, OUTPUT);
   Serial.print(touch);
if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
  p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
  p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
  // Touch area for box 1
  // Touch area for box 2
  if (p.x > 0 && p.x < 240) {
   if (p.y > 0 && p.y < 120) {
//    play_pause=!play_pause;
    touch = 2;
```

```
  }
 }
  // Touch area for box 3
 if (p.x > 0 && p.x < 120) {
 if (p.y > 120 && p.y < 240) {
  touch = 3;
  if(play_pause){
  if(lol<4){
  lol=lol+1;
  }else {
   lol=1;
  }
  }
 }
}
// Touch area for box 4
 if (p.x > 120 && p.x < 240) {
  if (p.y > 120 && p.y < 240) {
   touch = 4;
   if(play_pause){
   if(lol!=1){
```

```
   lol=lol-1;
   }
   else {
    lol=4;
   }
  }
  }
 }
}
// Process the touch in box 2
if (touch == 2) {
//digitalWrite(LED_PIN, LOW);
 tft.setCursor(100,250);
 tft.print(lol);
play_pause=!play_pause;
 delay(100);
}
  if (touch == 3) {
 digitalWrite(LED_PIN, LOW);
tft.setCursor(100,250);
tft.print(lol);
```

```
  delay(100);
 }
   if (touch == 4) {
  digitalWrite(LED_PIN, LOW);
 tft.setCursor(100,250);
tft.print(lol);
  delay(100);
}
}
void requestEvent(){
char character = touch ;
Wire.write(character ) ;
touch=0;
//play_pause=true;
}
```


fritzing
TEMP