**SİSTEM ANALİZİ VE TASARIMI DERSİ PROJESİ**

|  |  |
| --- | --- |
| **Öğrenci 1-Adı ve Soyadı-Numarası:** | **Abdulkadir Can Kinsiz-18811200002** |
| **Öğrenci 2-Adı ve Soyadı-Numarası:** |  |

* **Planlama ve Giriş Aşaması**

|  |  |
| --- | --- |
| *Projenin Başlığı* | **Akıllı Servis** |
| *Projenin Amacı* | **Servislerde hareket esnasında çocukların kemer takıp takmadığını göstererek can kaybını engellemek.** |
| *Hangi alana ait bir proje olacağı (web, oyun, masaüstü vb.)* | **Mobil** |
| *Hangi yazılımlardan faydalanılacağı* | **Android Studio - Arduino** |
| *Projede görev alacak olan kişilerin görevleri (Ad-soyad ve hangi işi yapacağı)* | **Abdulkadir Can Kinsiz-Android ve Arduino** |

* **Geliştirme ve zenginleştirme Aşaması**

|  |  |
| --- | --- |
| *Projenin seçilmesinin nedenleri (Neden bu projeye ihtiyaç var? Gerekçeleri nelerdir?)* | **Şoförler ve sürücüler, araçlarındaki kişilerin kemer takıp takmadığını görerek can kaybını azaltabilir.** |
| *Projenin hitap edeceği insan grubu (ör: ilkokul düzeyinde öğrencilerin oynayabileceği bir oyun geliştirmek)* | **Şofor ve sürücüler.** |
| *Proje yer alacak olan başlıklar, unsurlar, bileşenler (örn: öğretmen bilgileri, öğrenci bilgileri, not bilgileri, devamsızlık bilgileri vb.)* | **Emniyet Kemeri Değerlerini Okuma, Koltuk Değerlerini Okuma, Android Yazılımı** |

* **Tasarım, Algoritmik Akış ve Kodlar**

* **Emniyet Kemeri Değerlerini Okuma İşlemleri**

|  |  |
| --- | --- |
| Kodları | Algoritmik Açıklama |
| int fPin = 22;  int lPin = 44;  int seatsNumber = lPin - fPin + 1;  String seats[seatsNumber];  void setup() {  pinMode(2, OUTPUT);//Kemer çıkış pinimi 2 olarak seçtim ve pin modunu belirttim  digitalWrite(2,1);//Çıkış pinimden sürekli devam eden bir sinyal gönderdim  for (int i = fPin-2; i <= lPin; i++)  pinMode(i, INPUT);//Kemer giriş pinlerinin pin modunu belirttim    } | **Sensör olarak kendi yaptığım basınç sensörü ve emniyet kemerini kullanacağım. Emniyet kemerleri kemer takıldığında tıpkı bi lamba anahtarı gibi bir devreyi kapatır. Devre kapalı olunca sinyal geçer ve sorun görülmez. Sinyal yok ise sorun görülür ve kullanıcıya bildirilir. Emniyet kemeri devresi 2 kablodur. Birinden sinyal çıkar diğerinden yakalanır. Ben bütün kemerlerin çıkış sinyallerini tek bir Arduino pinine bağladım. giriş pinlerini ise başlangıç pini ve bitiş pini olarak 2 değişkende tutarak bir döngü yardımıyla bu iki pin arasında kalan bütün pinleri okudum. Daha sonradan koltuk verilerini göndereceğimden bir dizi tanımladım.** |
|  | |

* **Koltuk Değerlerini Okuma İşlemleri**

|  |  |
| --- | --- |
| **Kodları** | **Açıklaması** |
| int readSensor(uint8\_t senseAndChargePin)  {  uint8\_t myPin\_mask = digitalPinToBitMask(senseAndChargePin);  volatile uint8\_t \*myPin\_port = portInputRegister(digitalPinToPort(senseAndChargePin));  int left = TIMEOUT\_ATTEMPTS;  noInterrupts();  pinMode(senseAndChargePin, INPUT\_PULLUP);  do  {  left--;  } while (((\*myPin\_port & myPin\_mask) == 0) && left>0);    interrupts();  pinMode(senseAndChargePin, INPUT); //charge durdur  int roundsMade = TIMEOUT\_ATTEMPTS - left;    delayMicroseconds(roundsMade \* DISCHARGE\_FACTOR);    return roundsMade;  } | **Koltuklarda oturan birisinin olup olmadığını anlamam için kendi ürettiğim temel kapasitör prensibini temel alan bir basınç sensörünü kullandım. Bağlantı şemasını basitçe anlatacak olursam bir adet föyün içinde bir adet a4 kağıt ve kağıdın her bir yüzüne yüzey alanının %80'i büyüklüğünde aluminyum folyo yapıştırılıp her 2 folyoya kablo bağlayıp kablolardan birini Arduino GND pinine diğerini ise değer okuyacağımız pine bağlıyoruz. Föyün hava akışını bant ile kapatıyoruz(Föy yerine hava geçirmez poşetler kullanılabilir). Çalışma mantığını ise basitçe şöyle anlatayım; içerideki sıkışan hava üzerine ne kadar basınç uygulanırsa hava o kadar sıkışıyor ve aluminyum folyo üzerindeki elektronlar sıkışan hava ile daha hızlı ilerlemeye başlıyor. Bizde Arduino kısmında gelen sinyale karşı bir sinyal gönderip gelen sinyali çok kısa süreliğine durdurup tekrardan gönderdiğimiz sinyali kesiyoruz ve ilk aldığımız sinyal ile bir sonraki sinyal arasındaki süre farkıyla bazı hesaplamalar yapıp bir değer elde ediyoruz. Bu değer bizim ağırlığımız oluyor. Yeterince kalibre edilebilirse insan kilosuna yakın değerler elde edilebiliyor. Tabi biz uygulamamızda bu değerleri yuvarlayıp daha basit değerler elde ediyoruz. Değer okuyan fonksiyonumuz yan taraftadır.** |

* **Android Yazılımı İşlemleri**

|  |  |
| --- | --- |
| Tasarım | Algoritmik Açıklama |
|  | **Android tarafında basitçe algoritması Arduino'dan gelen değerleri bluetooth ile okumak ve bunları tasarımla birleştirmek. Android'de bluetooth ile (uzun) değer okumanın en sağlıklı yolu buffer kullanmaktır. Bende projemde öyle kullandım. Tasarım kısmında ise kullanıcın kendi aracını kendi istediği şekilde tasarlayabilmesini sağlamak istedim ve onun için düzenleme kısmı ekleyip orada koltuk ve koridor ekleme butonları ekledim. Yan tarafta tasarım kısmını ve aşağıda ana işlemlerin ve tasarımın yapıldığı kod kısımlarını görebilirsiniz.** |
| Kodları | |
| import 'dart:async'; import 'dart:convert'; import 'dart:io'; import 'dart:math'; import 'dart:typed\_data'; import 'package:flutter/material.dart'; import 'package:flutter/services.dart'; import 'package:flutter/rendering.dart'; import 'package:flutter\_bluetooth\_serial/flutter\_bluetooth\_serial.dart'; import 'package:flutter\_bluetooth\_serial\_example/SelectBondedDevicePage.dart'; import 'package:path\_provider/path\_provider.dart';  List<String> koltukid = new List(); List<String> kemer = new List(); List<String> koltuk = new List(); List<int> koltukmax = new List(); String textfromarduino = ""; int \_selectedIndex = 0;  class ChatPage extends StatefulWidget {  final BluetoothDevice server;  const ChatPage({this.server});   @override  \_ChatPage createState() => new \_ChatPage(); }  class \_Message {  int whom;  String text;   \_Message(this.whom, this.text); }  bool q = false; Future<void> reRead() async {  FileUtils.*readFromFile*().then((contents) {  if (contents.toString().length > 2) {  var koltuk\_koridor\_tasarimi = contents.split(",");  if (koltuk\_koridor\_tasarimi.length > 0) {  koltukid.clear();  koltuk.clear();  kemer.clear();  for (int i = 0; i < koltuk\_koridor\_tasarimi.length; i++) {  if (koltuk\_koridor\_tasarimi[i].toString() != 'koridor') {  var infos = koltuk\_koridor\_tasarimi[i].toString().split("\_");  koltukid.add(infos[1]);  kemer.add(infos[2]);  koltuk.add(infos[3]);  } else {  if (koltuk\_koridor\_tasarimi[i].toString() == 'koridor') {  koltukid.add("koridor");  kemer.add('koridor');  koltuk.add('koridor');  }  }  }  }  } else {  if (!q) {  q = true;  FileUtils.*saveToFile*("k\_0\_0\_0,");  reRead();  } else  reRead();  }  }); }  var r\_container = new Container(  color: Colors.*red*,  padding: const EdgeInsets.all(8),  child: IconButton(icon: Icon(Icons.*airline\_seat\_recline\_normal*), onPressed: () {}), );  var g\_container = new Container(  color: Colors.*green*,  padding: const EdgeInsets.all(8),  child: IconButton(icon: Icon(Icons.*airline\_seat\_recline\_normal*), onPressed: () {}), );  var y\_container = new Container(  color: Colors.*yellow*,  padding: const EdgeInsets.all(8),  child: IconButton(icon: Icon(Icons.*airline\_seat\_recline\_normal*), onPressed: () {}), );  var b\_container = new Container(  color: Colors.*blue*,  padding: const EdgeInsets.all(8),  child: IconButton(icon: Icon(Icons.*airline\_seat\_recline\_normal*), onPressed: () {}), );  var nullContainer = new Container(  color: Colors.*transparent*,  padding: const EdgeInsets.all(8), ); var kor\_Container = new Container(  color: Colors.*deepOrange*,  padding: const EdgeInsets.all(8), );  class FileUtils {  static Future<String> get *getFilePath* async {  final directory = await getApplicationDocumentsDirectory();  return directory.path;  }   static Future<File> get *getFile* async {  final path = await *getFilePath*;  return File('$path/realTest.txt');  }   static Future<File> *saveToFile*(String data) async {  final file = await *getFile*;  FileUtils.*readFromFile*().then((contents) {  return file.writeAsString(contents + data);  });  }   static Future<void> *del*() async {  final file = await *getFile*;  await file.delete();  }   static Future<void> *crt*() async {  final file = await *getFile*;  await file.create();  }   static Future<String> *readFromFile*() async {  try {  final file = await *getFile*;  String fileContents = await file.readAsString();  return fileContents;  } catch (e) {  return "";  }  } }  class \_ChatPage extends State<ChatPage> {  static final *clientID* = 0;  static final *maxMessageLength* = 4096 - 3;  BluetoothConnection connection;   List<\_Message> messages = List<\_Message>();  String \_messageBuffer = '';   final TextEditingController textEditingController =  new TextEditingController();  final ScrollController listScrollController = new ScrollController();   bool isConnecting = true;  bool get isConnected => connection != null && connection.isConnected;   bool isDisconnecting = false;   @override  void initState() {  super.initState();  BluetoothConnection.toAddress(widget.server.address).then((\_connection) {  print('Connected to the device');  connection = \_connection;  setState(() {  isConnecting = false;  isDisconnecting = false;  });   connection.input.listen(\_onDataReceived).onDone(() {  if (isDisconnecting) {  print('Disconnecting locally!');  } else {  print('Disconnected remotely!');  }  if (this.mounted) {  setState(() {});  }  });  }).catchError((error) {  print('Cannot connect, exception occured');  print(error);  });  }   @override  void dispose() {  // Avoid memory leak (`setState` after dispose) and disconnect  if (isConnected) {  isDisconnecting = true;  connection.dispose();  connection = null;  }   super.dispose();  }   @override  Widget build(BuildContext context) {  final List<Row> list = messages.map((\_message) {  if (textfromarduino != messages.last.text.trim()) {  print("sea:"+messages.last.text.trim());  UpdateSeat(messages.last.text.trim());  textfromarduino = messages.last.text.trim();  } else {}  return Row(  children: <Widget>[  Container(  child: Text(  (text) {  return text == '/shrug' ? '¯\\\_(ツ)\_/¯' : text;  }(\_message.text.trim()),  style: TextStyle(color: Colors.*white*)),  padding: EdgeInsets.all(12.0),  margin: EdgeInsets.only(bottom: 8.0, left: 8.0, right: 8.0),  width: 222.0,  decoration: BoxDecoration(  color:  \_message.whom == *clientID* ? Colors.*blueAccent* : Colors.*grey*,  borderRadius: BorderRadius.circular(7.0)),  ),  ],  mainAxisAlignment: \_message.whom == *clientID* ? MainAxisAlignment.end  : MainAxisAlignment.start,  );  }).toList();  reRead();  startContact();  return Scaffold(  bottomNavigationBar: BottomNavigationBar(  type: BottomNavigationBarType.fixed,  items: const <BottomNavigationBarItem>[  BottomNavigationBarItem(  icon: Icon(Icons.*airline\_seat\_recline\_normal*,color: Colors.*red*,),  backgroundColor: Colors.*red*,  title: Text('Boş',style: TextStyle(color: Colors.*blue*)),  ),  BottomNavigationBarItem(  icon: Icon(Icons.*airline\_seat\_recline\_normal*,color: Colors.*yellow*),  backgroundColor: Colors.*yellow*,  title: Text('Kemersiz',style: TextStyle(color: Colors.*blue*)),  ),  BottomNavigationBarItem(  icon: Icon(Icons.*airline\_seat\_recline\_normal*,color: Colors.*blue*),  backgroundColor: Colors.*blue*,  title: Text('Kemerli',style: TextStyle(color: Colors.*blue*)),  ),  BottomNavigationBarItem(  backgroundColor: Colors.*green*,  icon: Icon(Icons.*airline\_seat\_recline\_normal*,color: Colors.*green*),  title: Text('Dolu',style: TextStyle(color: Colors.*blue*),),  ),  ],  ),  appBar: AppBar(  automaticallyImplyLeading: false,  title: (isConnecting  ? Text('Connecting chat to ' + widget.server.name + '...')  : isConnected  ? Text('Live chat with ' + widget.server.name)  : Text('Chat log with ' + widget.server.name)),  actions: <Widget>[  (IconButton(  icon: Icon(Icons.*settings*),  onPressed: () {  reRead();  Navigator.*of*(context).pushReplacement(new MaterialPageRoute(  builder: (BuildContext context) => SettingsPage()));  }))  ],  ),  body: Row(  children: <Widget>[  koltukGrid(),  ],  )   );  }  Widget koltukGrid() {  return Container(child:Expanded(  child: GridView.count(  primary: false,  padding: const EdgeInsets.all(20),  crossAxisSpacing: 5,  mainAxisSpacing: 5,  crossAxisCount: 5,  children: <Widget>[  for (int i = 0; i < koltukid.length; i++)  if ('koridor' != koltuk[i].toString())  if (koltuk[i].toString() != '0')  if (kemer[i].toString() != '0') g\_container else y\_container  else if (kemer[i].toString() != '0')  b\_container  else  r\_container  else  nullContainer  ],  ),  )  );  }   Widget infobarGrid() {  return Container(  child:Expanded(  child: GridView.count(  childAspectRatio:4,  primary: false,  padding: const EdgeInsets.all(10),  crossAxisCount: 4,  children: <Widget>[  Text( 'Boş',style: TextStyle(fontSize: 21,color: Colors.*black54*)),  Text( 'Dolu',style: TextStyle(fontSize: 21,color: Colors.*greenAccent*)),  Text( 'Kemersiz',style: TextStyle(fontSize: 21,color:Colors.*yellow*)),  Text( 'Kemerli',style: TextStyle(fontSize: 21,color:Colors.*blue*)),  ],  ),  )  );  }    Future<void> startContact() {  Timer.periodic(Duration(seconds: 10), (timer) {  \_sendMessage("x");  print("X:");  });  }   Future<void> \_onDataReceived(Uint8List data) async {  // Allocate buffer for parsed data  int backspacesCounter = 0;  data.forEach((byte) {  if (byte == 8 || byte == 127) {  backspacesCounter++;  }  });  Uint8List buffer = Uint8List(data.length - backspacesCounter);  int bufferIndex = buffer.length;   // Apply backspace control character  backspacesCounter = 0;  for (int i = data.length - 1; i >= 0; i--) {  if (data[i] == 8 || data[i] == 127) {  backspacesCounter++;  } else {  if (backspacesCounter > 0) {  backspacesCounter--;  } else {  buffer[--bufferIndex] = data[i];  }  }  }   // Create message if there is new line character  String dataString = String.fromCharCodes(buffer);   int index = buffer.indexOf(13);  if (~index != 0) {  // \r\n  setState(() {  messages.add(\_Message(  1,  backspacesCounter > 0  ? \_messageBuffer.substring(  0, \_messageBuffer.length - backspacesCounter)  : \_messageBuffer + dataString.substring(0, index)));   \_messageBuffer = dataString.substring(index);  });  } else {  \_messageBuffer = (backspacesCounter > 0  ? \_messageBuffer.substring(  0, \_messageBuffer.length - backspacesCounter)  : \_messageBuffer + dataString);  }  }   Future<void> \_sendMessage(String text) async {  text = text.trim();  textEditingController.clear();   if (text.length > 0) {  try {  connection.output.add(utf8.encode(text));  await connection.output.allSent;  setState(() {  messages.add(\_Message(*clientID*, text));  });   Future.delayed(Duration(milliseconds: 333)).then((\_) {  listScrollController.animateTo(  listScrollController.position.maxScrollExtent,  duration: Duration(milliseconds: 333),  curve: Curves.*easeOut*);  });  } catch (e) {  // Ignore error, but notify state  setState(() {});  }  }  } }  Future<void> UpdateSeat(String a) async {  List seats = a.split(",");  for (int i = 0; i < seats.length; i++) {  if (seats[i].toString() == "") {  } else {  List infos = seats[i].split("\_");  for (int i = 0; i < koltukid.length; i++) {  if (koltukid[i] == infos[1]) {  kemer[i] = infos[2];  koltuk[i] = infos[3];  }  }  }  }   UpdateTheFile(); }  Future<void> UpdateTheFile() async {  String texttofile = "";   for (int i = 0; i < koltukid.length; i++) {  if (koltuk[i] != 'koridor')  texttofile = texttofile +  "k\_" +  koltukid[i].toString() +  "\_" +  kemer[i].toString() +  "\_" +  koltuk[i].toString() +  ",";  else  texttofile = texttofile + 'koridor,';  }  await FileUtils.*del*();  await FileUtils.*crt*();  await FileUtils.*saveToFile*(texttofile);  await reRead();  print("txt:" + texttofile);  //FILE SAVE }  class SettingsPage extends StatelessWidget {  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  automaticallyImplyLeading: false,  title: (Text('Koltuk Düzeni')),  leading: IconButton(  color: Colors.*green*,  icon: Icon(Icons.*refresh*),  onPressed: () {  reRead();  Navigator.*of*(context).pushReplacement(new MaterialPageRoute(  builder: (BuildContext context) => SettingsPage()));  }),   actions: <Widget>[  Row(  children: <Widget>[  IconButton(  color: Colors.*green*,  icon: Icon(Icons.*clear\_all*),  onPressed: () {  FileUtils.*del*();  q=false;  koltukid.clear();  kemer.clear();  koltuk.clear();  FlutterBluetoothSerial.instance.requestDisable();   Future.delayed(const Duration(milliseconds: 500), () {  FlutterBluetoothSerial.instance.requestEnable();   Future.delayed(const Duration(milliseconds: 500), () {  FlutterBluetoothSerial.instance  .onStateChanged()  .listen((BluetoothState state) {  FlutterBluetoothSerial.instance  .onStateChanged()  .listen((BluetoothState state) {  Navigator.*pushAndRemoveUntil*(  context,  MaterialPageRoute(  builder: (BuildContext context) =>  SelectBondedDevicePage(  checkAvailability: false),  ),  ModalRoute.*withName*('/'));  });  });  });  });  }),  IconButton(  color: Colors.*red*,  icon: Icon(Icons.*airline\_seat\_recline\_normal*),  onPressed: () {  koltukmax.clear();  for(int j=0;j<koltukid.length;j++){  if(koltukid[j]!="koridor")  koltukmax.add(int.*parse*(koltukid[j]));}   int id = koltukmax.reduce(max) + 1;  String strid = id.toString();  FileUtils.*saveToFile*("k\_" + strid + "\_0\_0,");  Future.delayed(const Duration(milliseconds: 100), () {  reRead();  Navigator.*of*(context).pushReplacement(  new MaterialPageRoute(  builder: (BuildContext context) =>  SettingsPage()));  });  }),  IconButton(  color: Colors.*green*,  icon: Icon(Icons.*add\_box*),  onPressed: () {  FileUtils.*saveToFile*("koridor,");  Future.delayed(const Duration(milliseconds: 100), () {  reRead();  Navigator.*of*(context).pushReplacement(  new MaterialPageRoute(  builder: (BuildContext context) =>  SettingsPage()));  });  }),  IconButton(  color: Colors.*green*,  icon: Icon(Icons.*backspace*),  onPressed: () {  koltukid.clear();  kemer.clear();  koltuk.clear();  FlutterBluetoothSerial.instance.requestDisable();   Future.delayed(const Duration(milliseconds: 500), () {  FlutterBluetoothSerial.instance.requestEnable();   Future.delayed(const Duration(milliseconds: 500), () {  FlutterBluetoothSerial.instance  .onStateChanged()  .listen((BluetoothState state) {  FlutterBluetoothSerial.instance  .onStateChanged()  .listen((BluetoothState state) {  Navigator.*pushAndRemoveUntil*(  context,  MaterialPageRoute(  builder: (BuildContext context) =>  SelectBondedDevicePage(  checkAvailability: false),  ),  ModalRoute.*withName*('/'));  });  });  });  });  }),  ],  ),  ],  ),  body: GridView.count(  primary: false,  padding: const EdgeInsets.all(20),  crossAxisSpacing: 10,  mainAxisSpacing: 10,  crossAxisCount: 5,  children: <Widget>[  for (int i = 0; i < koltukid.length; i++)  if ('koridor' != koltuk[i].toString())  if (koltuk[i].toString() != '0')  if (kemer[i].toString() != '0') g\_container else y\_container  else if (kemer[i].toString() != '0')  b\_container  else  r\_container  else  kor\_Container  ],  ),  );  } }  bool ArrayContains(int data, List searchArray) {  for (int i = 0; i < searchArray.length; i++) {  if (data == searchArray[i]) return true;  }  return false; }  //DOSYA YAZMA | |

* **Değerlendirme**

Benzer projeler var fakat ben bu projede fiyatı minimuma düşürerek daha farklı bir proje ortaya çıkardım. Buna benzer projelerden biri olan araç takip sistemleri piyasada şuan 3000 TL gibi bir fiyattan satılıyor. Benim projemde ise fiyat 500 TL'nin altında. Projenin araç içinde saklanacağı bir kutu tasarlamayı düşünüyorum soketler sayesinde sensörlerin bu kutuya takılıp arıza tespiti ve onarımı gibi durumlarda kolaylık sağlayacaktır. Bu projeyi tamamen bitirip Android ve Arduino kısmını github üzerinden paylaşacağım.

|  |  |
| --- | --- |
| **I-** | **Amacına uygun oldu mu ?**  Amacına tam olarak uygun oldu. Tek eksiği yukarıda bahsettiğim gibi bir kutu tasarlayıp bütün her şeyi kutu içine toparlamam gerekmesi. |
| **II-** | **Yaşadığım Sorunlar ve kullandığım çözümler**  Bluetooth üzerinden uzun mesajlar gönderirken mesajın parazite uğrayıp hasar alması gibi bir sorun vardı, Buffer kullanarak çözdüm. Arduino kısmında ağırlık sensörü gibi parçalar kullandım fakat istediğim değerleri göstermedi. Kendi ağırlık sensörümü tasarladım. |