Практическое занятие №12

Цели занятия

- Изучить архитектуру и возможности аппаратной части мобильных устройств.
- Ознакомиться с API камеры и галереи во Flutter.
- Научиться создавать приложения, использующие камеру и хранилище устройства.
- Разобраться с разрешениями, обработкой изображений и сохранением данных.

Теоретическая часть

Аппаратная часть мобильных устройств

Современные смартфоны представляют собой компактные вычислительные системы, включающие:

Компонент	Назначение
Процессор (CPU)	Выполняет вычисления, обработку данных, управление задачами.
Графический процессор (GPU)	Обрабатывает изображения, видео, 3D-графику, рендеринг интерфейса.
Оперативная память (RAM)	Хранит данные временно во время работы приложений.
Постоянная память (Storage)	Содержит систему, приложения и пользовательские данные.
Дисплей	Основной интерфейс взаимодействия пользователя с системой.
Сенсорный экран	Обеспечивает ввод касаниями.
Камера (фронтальная, основная)	Позволяет получать изображение с матрицы и использовать в приложениях.
Датчики (сенсоры)	Акселерометр, гироскоп, датчик приближения, освещенности, GPS и др.
Коммуникации	Wi-Fi, Bluetooth, NFC, мобильные сети.

Для разработчика важно:

Каждый аппаратный модуль доступен через системные API. В Flutter это реализовано с помощью плагинов (`camera`, `image_picker`, `permission_handler` и др.).

Доступ к аппаратным ресурсам в Flutter

Flutter не работает напрямую с оборудованием. Для этого используются **плагины**, которые оборачивают нативные API Android/iOS.

- **image_picker** — сделать фото или выбрать из галереи.

- **camera** — получить прямой доступ к камере (предпросмотр, управление вспышкой, запись видео).
- **permission_handler** — запрашивать системные разрешения (камера, хранилище, микрофон и др.).
- **path_provider** — сохранять файлы в хранилище устройства.

Разрешения (Permissions)

Для работы с камерой или файлами необходимо запросить разрешение у пользователя.

Android:

```
android/app/src/main/AndroidManifest.xml
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

iOS:

```
ios/Runner/Info.plist
<key>NSCameraUsageDescription</key>
<string>Для работы приложения требуется доступ к камере</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>Для выбора изображений из галереи требуется
разрешение</string>
```

Работа с камерой и галереей (Flutter)

1) Быстрый вариант — `image_picker`

Позволяет сделать фото или выбрать из галереи:

```
dependencies:
  image_picker: ^1.1.1
  permission_handler: ^11.3.0
import 'package:image_picker/image_picker.dart';

final picker = ImagePicker();
```

```
    final XFile? photo = await picker.pickImage(source: ImageSource.camera);
```

Результат — объект `XFile`, у которого можно получить путь к файлу `photo.path`.

2) Продвинутый вариант — `camera`

Позволяет создать **предпросмотр**, переключать камеры, управлять вспышкой, записывать видео.

```
    dependencies:
      camera: ^0.11.0
      path_provider: ^2.1.4
```

Поток камеры

1. Получить список доступных камер.
2. Инициализировать `CameraController`.
3. Вывести `CameraPreview`.
4. Сделать снимок методом `takePicture()`.

Обработка и сохранение изображений

Для работы с файлами:

- `path_provider` — пути к временным и постоянным директориям.
- `dart:io` — чтение/запись.
- `image` — библиотека для фильтров, кропа, поворота и пр.

Рекомендации по UX

- Перед съемкой показывать **предпросмотр**.
- После съемки — экран подтверждения или предпросмотра.
- Добавлять **индикацию** (вспышка, фокус, таймер).
- Сообщать пользователю о сохранении и пути к файлу.
- Учитывать ориентацию экрана.

Практическая часть

Задание

Создать простое приложение **«CameraApp»**, которое:

1. Показывает кнопки **Сделать фото** и **Выбрать из галереи**.
2. После выбора — отображает фото на экране.
3. (Дополнительно) Позволяет сохранить фото в память устройства.

Подготовка проекта

```
flutter create camera_app
cd camera_app
```

Установка зависимостей

`pubspec.yaml`:

```
dependencies:
  flutter:
    sdk: flutter
  image_picker: ^1.1.1
  permission_handler: ^11.3.0
  path_provider: ^2.1.4
```

Настройка разрешений

Добавьте соответствующие строки в **AndroidManifest.xml** и **Info.plist** (см. выше).

Основной экран приложения

`lib/main.dart`:

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:permission_handler/permission_handler.dart';
import 'package:path_provider/path_provider.dart';

void main() => runApp(const CameraApp());
```

```
class CameraApp extends StatelessWidget {
  const CameraApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Camera Demo',
      theme: ThemeData(useMaterial3: true),
      home: const CameraPage(),
    );
  }
}

class CameraPage extends StatefulWidget {
  const CameraPage({super.key});

  @override
  State<CameraPage> createState() => _CameraPageState();
}

class _CameraPageState extends State<CameraPage> {
  File? _image;
  final picker = ImagePicker();

  Future<void> _checkPermissions() async {
    await Permission.camera.request();
    await Permission.photos.request();
  }

  Future<void> _getImage(ImageSource source) async {
    await _checkPermissions();
    final XFile? pickedFile = await picker.pickImage(source:
source);
    if (pickedFile != null) {
      setState(() => _image = File(pickedFile.path));
    }
```

```
      }

      Future<void> _saveImage() async {
        if (_image == null) return;
        final dir = await getApplicationDocumentsDirectory();
         final newFile = await _image!.copy('${dir.path}/photo_$
{DateTime.now().millisecondsSinceEpoch}.jpg');
        ScaffoldMessenger.of(context).showSnackBar(SnackBar(cont
ent: Text('Сохранено: ${newFile.path}')));
      }

      @override
      Widget build(BuildContext context) {
        return Scaffold(
          appBar: AppBar(title: const Text('Работа с камерой')),
          body: Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                _image == null
                    ? const Text('Фото не выбрано')
                       : Image.file(_image!, width: 250, height:
250, fit: BoxFit.cover),
                const SizedBox(height: 20),
                ElevatedButton.icon(
                                          onPressed:   ()   =>
_getImage(ImageSource.camera),
                  icon: const Icon(Icons.camera_alt),
                  label: const Text('Сделать фото'),
                ),
                ElevatedButton.icon(
                                          onPressed:   ()   =>
_getImage(ImageSource.gallery),
                  icon: const Icon(Icons.image),
                  label: const Text('Выбрать из галереи'),
                ),
```

```
              const SizedBox(height: 10),
              ElevatedButton.icon(
                onPressed: _saveImage,
                icon: const Icon(Icons.save),
                label: const Text('Сохранить фото'),
              ),
            ],
          ),
        ),
      );
  }
}
```

Контрольные точки

1. Приложение запускается без ошибок.
2. При нажатии “Сделать фото” открывается камера.
3. После съемки фото отображается на экране.
4. При нажатии “Выбрать из галереи” можно выбрать изображение.
5. Кнопка “Сохранить фото” копирует файл в локальное хранилище.
6. Пользователь получает уведомление о сохранении.

Что сдаётся (отчёт)

- Скриншот работы приложения:
  1. Интерфейс (главный экран).
  2. Камера в действии.
  3. Отображение выбранного фото.
- Краткое описание шагов реализации.
- Ключевые фрагменты кода.
- Заключение (что нового узнали, какие были трудности).

Требования к отчёту

- 2–4 страницы, формат `.docx`.

- Структура:
    1. Цели работы
    2. Ход выполнения (установка зависимостей, разрешения, код)
    3. Скриншоты
    4. Вывод

Контрольные вопросы

1. Какие аппаратные компоненты смартфона можно использовать в приложениях Flutter?
2. Какие библиотеки используются для работы с камерой?
3. Чем отличается `camera` от `image_picker`?
4. Какие разрешения нужно добавить в Android и iOS?
5. Как реализовать сохранение снимков на устройство?

Дополнительно (+1 балл)

- Добавьте возможность **снимать видео** и воспроизводить его в приложении.
- Реализуйте **фильтры или обработку** изображения (черно-белый, размытие и т.д. — библиотека `image`).
- Покажите **предпросмотр** фото перед сохранением.
- Добавьте **вспышку/переключение камер** (через пакет `camera`).