1. IZPIT

1. (11 points) Arduino programming.
   (a) (2 points) What does the listed code do?
   (b) (5 points) Modify the code so that the delay is not fixed, but set to a value between 0 and 1023 ms according to the potentiometer connected to pin A0.
   (c) (2 points) Describe the difference between analog and digital input pins on Arduino UNO;
   (d) (2 points) Describe the difference between analog and digital output pins on Arduino UNO;

```
1  int pinLED=9;
2  int controlPin=A0;
3
4  void setup() {
5    Serial.begin(9600);
6    pinMode(pinLED, OUTPUT);
7  }
8
9  void loop() {
10
11   digitalWrite(pinLED, HIGH);
12
13   delay(1000);
14
15   digitalWrite(pinLED, LOW);
16
17   delay(1000);
18
19 }
```

a) LED flashes (turn on, 1 s, turn off, 1 s,...)

b)

```
int pinLED = 9;
int controlPin = A0;

int potentiometer = 0;
void setup(){
Serial.begin(9600);
pinMode(pinLED, OUTPUT);
pinMode(controlPin, INPUT);
}

void loop(){
potentiometer = analogRead(controlPin);
digitalWrite(pinLED, HIGH);
delay(potentiometer);
digitalWrite(pinLED, LOW);
delay(potentiometer);
}
```

c) Analog Input Pins: These pins can measure voltage levels ranging from 0 to 5 volts. They can provide a resolution of 1024 distinct values (0 to 1023) representing the input voltage.
Digital Input Pins: Digital input pins can only read two states: HIGH or LOW. They are typically used to detect the state of a switch or a sensor that provides a binary (on/off) signal. Digital input pins are also compatible with 5-volt logic signals.

d) Isti shit sam d ma analog 0 do 255

2. (12 points) Hybrid application development.
   (a) (3 points) What is Hybrid application development? Why is it useful?
   (b) (3 points) Describe the functionality of Cordova "plugins".
   (c) (4 points) What are possible drawbacks of Hybrid development? How can you minimize them?
   (d) (2 points) When developing an Android app you might want to implement something when the corresponding Activity is paused. To do that, you simply override onPause and put your code in this callback. How can you implement the same functionality if you are developing a hybrid app using Cordova?

a) Hybrid application development refers to the process of creating mobile applications using web technologies such as HTML, CSS, and JavaScript, which are then wrapped in a native container. This approach allows developers to build applications that can run on multiple platforms, including iOS, Android, and others. Hybrid development combines the benefits of web development with the ability to access device features and APIs through native wrappers.
The key advantage of hybrid application development is its cross-platform compatibility. By using a single codebase, developers can target multiple platforms, saving time and effort compared to building separate native applications for each platform. Hybrid apps also have access to device features, such as camera, accelerometer, and geolocation, through various frameworks and plugins.

b) Cordova plugins are add-ons or extensions that enhance the functionality of hybrid applications built with Apache Cordova (formerly known as PhoneGap). Cordova provides a set of core APIs that allow access to native device features, but plugins extend this functionality by providing additional APIs for specific features or services.
Cordova plugins can be developed by the community or by third-party developers and can cover a wide range of functionalities, such as accessing the device's camera, reading contacts, playing audio, interacting with the file system, integrating with social media platforms, and much more. These plugins provide a bridge between the JavaScript code in the hybrid app and the native APIs of the underlying platform, enabling developers to leverage native features seamlessly.

c) Possible drawbacks of hybrid development include:

- Performance: Hybrid apps may not perform as well as native apps, especially for graphics-intensive or processor-intensive tasks. The overhead of the web-based rendering engine and the bridge to native APIs can impact performance.
- User Experience: Hybrid apps may not provide the same level of user experience as native apps. They may feel less responsive or lack the native look and feel, which can affect user satisfaction.
- Limited Access to Device APIs: Hybrid apps rely on plugins to access certain device features and APIs. If a required plugin is not available or poorly maintained, it may limit the app's capabilities.
To minimize these drawbacks, consider the following approaches:

- Optimize Performance: Employ techniques such as optimizing JavaScript code, using efficient CSS and HTML, and leveraging hardware acceleration to enhance app performance.
- Utilize Native Components: Incorporate native components or frameworks where necessary to provide a more native-like experience and improved performance for specific features.
- Choose Plugins Carefully: Research and select well-maintained plugins with good community support to ensure access to required device APIs and functionality.

d) `document.addEventListener("pause", callback);`

3. (11 points) Kotlin.

   (a) (3 points) The following code is located in `MainActivity` of a programme that contains a `SimpleWorker` class for background data processing. The code does not compile as it contains errors and is also a mix of Kotlin and Java code:


```
1  // (as class properties) Kotlin code starts
2  val worker: SimpleWorker = SimpleWorker()
3  var handler: Handler
4  // Kotlin code ends
5
6  // (in MainActivity's onCreate) Java code starts
7  handler = new Handler(Looper.getMainLooper()) {
8      @Override
9      public void handleMessage(@NonNull Message msg) {
10         super.handleMessage(msg);
11         tvMessage.setText((String) msg.obj);
12     }
13 };
14 // Java code ends
```

   First, fix the bug in the Kotlin part of the code.

   (b) (3 points) Now, rewrite the Java code in Kotlin.

(c) (5 points) The following code located within MainActivity represents an AsyncTask for background processing. The task takes a parameter supplied when the task is executed, and calls a separate function `complexCalc` ten times, each time with the given parameter and an integer from 1 to 10. The UI contains a `progressBar` that should move from 0 to 100 (in 10-point increments, as the `complexCalc` is being called), it also contains a `statusText` TextView which should show "All Done!" when the processing is finished.

AsyncTaks is now deprecated. Rewrite the code using Kotlin coroutines instead. Make sure that the functionalities are preserved (i.e. background processing, UI updates):

```
1  // Task definition starts
2  inner class PostTask : AsyncTask<String, Int, String>() {
3
4      override fun onPreExecute() {
5          super.onPreExecute()
6
7          progressBar.visibility = View.VISIBLE
8          progressBar.progress = 0
9      }
10
11     override fun doInBackground(vararg strings: String?): String {
12
13         val myParam: String? = strings.get(0)
14
15         for (i in 1..10) {
16             try {
17                 complexCalc(myParam, i)
18             } catch (e: InterruptedException) {
19                 e.printStackTrace()
20             }
21             publishProgress(i*10)
22         }
23         return "All Done!"
24     }
25
26     override fun onProgressUpdate(vararg values: Int?) {
27         super.onProgressUpdate(*values)
28
29         progressBar.visibility = View.VISIBLE
30         progressBar.progress = values[0] ?: 0
31     }
32
33     override fun onPostExecute(result: String?) {
```



```
34         super.onPostExecute(result)
35         statusText.text = result
36         progressBar.visibility = View.GONE
37     }
38 }
39 // Task definition ends
40
41 // Task called from an onClick listener for a Button in MainActivity:
42 PostTask().execute("my favourite parameter")
```

a & b)

```
// (as class properties)
val worker: SimpleWorker = SimpleWorker()
lateinit var handler: Handler

// (in MainActivity's onCreate)
handler = object : Handler(Looper.getMainLooper()) {
    override fun handleMessage(msg: Message) {
        super.handleMessage(msg)
        tvMessage.text = msg.obj as String
    }
}
```

c)

```
val = scope = CoroutineScope(Dispatchers.Default)

// inside a button event listener:
scope.launch {
    withContext(Dispatchers.Default) {
        backgroundProcessing("my favourite parameter")
        for (i in 1..10) {
```

```
            try {
                complexCalc("my favourite parameter", i)
            } catch (e: InterruptedException) {
                e.printStackTrace()
            }
            progressBar.visibility = View.VISIBLE
            progressBar.progress = i * 10
        }
    }
    statusText.text = "All Done!"
    progressBar.visibility = View.GONE
}
```

4. (16 points) You are designing an Android app for booking a table in a restaurant. The app should get data about the restaurants from a remote REST server that has an endpoint allowing querying with geographic coordinates and a radius, returning a list of restaurants within a circle defined by the coordinates as the center and the radius. The server also has an endpoint for querying details of a particular restaurant ID. The server sends data in the JSON format.

   (a) (4 points) UI: Describe how you would implement the app so that it has two views – one for showing a gallery view of nearby restaurants and the other (opened once a user clicks on a selected restaurant) showing the details of the restaurant (menu, photos, etc.). Note: focus on the UI only, no need to provide location sensing details.

   (b) (4 points) Data management: Describe the data transfer/manipulation/storage elements of your app. The app should also provide a caching functionality, so that a user can browse the restaurants even if the connection is disrupted (of course, no booking is possible in this case). Fetching the data from a remote server, interfacing it with the UI elements, and long-term storage should all be clearly defined and described.

   (c) (4 points) Payment: When a user selects a restaurant where she would like to book a table, she needs put a small payment to secure the booking. Describe how you would implement the payment part of the app. Assume that the payment is handled through a third-party service (e.g. PayPal) and focus on reliability.

   (d) (4 points) Location determination: Explain how you would implement location sensing, so that the user's location is determined and used for fetching the restaurants within a $3km$ range when the app is opened. As a user is moving around while actively using the app, the gallery view should be periodically refreshed with a new set of nearby restaurants corresponding to the new location of the user.

   (e) (4 points) Bonus: When a user gets within a $300m$ radius from the restaurant she booked, the app should send a notification reminder to her phone. Describe how you would implement this functionality. It should work even when the app is not actively used and it should use the minimum amount of battery.

   Note: Please draw diagram(s) of your solution.

a) The app would have 1 Activity with a NavHostFragment and 2 Fragments; ListFragment that would contain would contain a RecyclerView to list of all restaurants (e.g. CardViews) and would use an Adapter would to connect the views with data objects. DetailsFragment would contain all the specific information regarding a restaurant and buttons to book it (ImageView, TextView,...) The 2 fragmants would be connected using the nav graph, IDs must be passed through so the right restaurant is displayed

b) The Fragments should obtain the data from ItemsViewModel, where the data is stored in LiveData objects. The ViewModel communicates to Repository class, which manages storing/fetching of data from REST API and the locat database. For the REST API, we will need Retrofit and define a DTO (we will also need GSON). For the local database, we will use Room to create it, then define the operations using DAO. The @Entity object Item will define a table of data in there. Once ran, the app pulls data from the server via the Repository, stores it in the database, and shows listings to the user via the Fragments that obtain the information from the ItemsViewModel.

c) Using WorkManager and the Repository class, we can request for data every 24 hours to check for any changes.

2. IZPIT

1. (10 points) The code shows an Arduino programme for reading temperature from a car's AC system digital temperature sensor and printing the temperature on the serial output.
   (a) (4 points) The air conditioner (AC) fan is controlled by a motor (connected to pin 11). The motor's rotation speed is proportional to the voltage supplied at its input pin. The fastest speed is achieved when 5V are at the input and then it linearly falls down as the voltage is decreased. The motor supports PWM input mode, and the board can output PWM on its pin 11 with ranges between 0 (always off) and 255 (always on). Expand the code so that the AC fan is turned on and operates with 25% speed if the temperature is between 25 and 28 degrees Celsius, and operates with 50% speed if the temperature is above 28 degrees Celsius.
   (b) (4 points) A seatbelt pretensioner ensures that when a crash happens, the seatbelts are tightly pulled in, so that the passengers are kept in the optimal position in their seats. The functioning of the pretensioner is explained in the figure – a chamber contains combustible gas, electrodes (activated by the microcontroller) create a spark when a crash is detected, the gas expands and creates pressure that pushes the piston and tightens the seatbelt. A crash sensor is connected to pin 3. This is a binary accelerometer that reports HIGH if the acceleration values are above the threshold that indicates a very sudden change (most likely a crash). Expand the code so that the pretensioner electrode (connected to pin 13) creates a spark (by setting pin 13 to HIGH) if a crash is detected. Crash detection and electrode activation should be handled by an interrupt service routine connected to crashSensorPin.
   (c) (2 points) What happens when an interrupt signal arrives on the same pin for which an Arduino programme is already executing the ISR at that moment?

a & b)

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

const int crashSensorPin = 3;       // the number of the crash sensor pin
const int pretensionerPin = 13;     // the number of the seatbelt pretensioner pin
```

```
const int ACfanPin = 11;            // the number of the AC fan pin

void setup() {
  pinMode(crashSensorPin, INPUT);
  pinMode(pretensionerPin, OUTPUT);
  pinMode(ACfanPin, OUTPUT);

  attachInterrupt(digitalPinToInterrupt(crashSensorPin), seatbelt, RISING);

  Serial.begin(9600);
  sensors.begin();
}

void loop() {
  // Temperature reading
  sensors.requestTemperatures(); // Send the command to get temperatures
  float temperature = sensors.getTempCByIndex(0);

  Serial.print("The temperature is: ");
  Serial.print(temperature);
  Serial.println(" degrees Celsius");

  // AC fan control (part a)
  if (temperature >= 25 && temperature <= 28) {
    analogWrite(ACfanPin, 64); // 25% speed-ish (or just do 0.25 * 255)
  } else if (temperature > 28) {
    analogWrite(ACfanPin, 128); // 50% speed-ish (or just do 0.5 * 255)
  } else {
    analogWrite(ACfanPin, 0); // Fan off
  }

  // Pretensioner electrode activation (part b)
  if (crashDetected) {
    digitalWrite(pretensionerPin, HIGH);
    delay(1000); // Electrode activation duration (1 second)
    digitalWrite(pretensionerPin, LOW); // Deactivate electrode
    crashDetected = false; // Reset crash detection
  }
}

void seatbelt(){
  digitalWrite(pretensionerPin, HIGH);
}
```

c) When an interrupt signal arrives on the same pin for which an ISR is already executing, it will wait for the currently executing ISR to finish before executing the new interrupt.

2. (12 points) WWW Platform.

   (a) (4 points) You are building a Voice-over-IP service for remote rural regions that connect to the Internet via satellite links. Briefly explain which transport-layer technology you would use and why.

   (b) (4 points) You are building an online social network Web client for remote rural regions. Explain how you would programme your solution to minimize the effect of TCP slow start over a satellite link.

   (c) (4 points) Your online social network Web client is using the traditional host-oriented addressing, which requires it to go to the same (set of) server(s) each time a certain content, e.g. a photo, is required. However, another endpoint in the village could have already downloaded the content. Propose a different content addressing scheme, that would be more bandwidth efficient in this case.

a) The most suitable transport-layer technology would be User Datagram Protocol (UDP), a connectionless transport protocol that provides a lightweight and efficient way to transmit data over the network.
In this scenario, the primary concern is real-time communication and minimizing latency. UDP offers lower overhead compared to Transmission Control Protocol (TCP) since it does not include mechanisms like reliable delivery, congestion control, and flow control. This lower overhead results in reduced latency and ensures faster transmission of voice packets.
Additionally, satellite links tend to have high latency and limited bandwidth, which can introduce delays and packet loss. UDP's connectionless nature allows it to better handle these conditions. It enables the VoIP application to send packets without waiting for acknowledgments, which can help mitigate the impact of latency and provide a smoother voice communication experience.
However, it's important to note that UDP does not provide built-in reliability or error recovery mechanisms. Therefore, the VoIP application would need to incorporate mechanisms such as packet loss detection, retransmission, and jitter buffering to ensure reliable voice transmission over the satellite link.

b) Connection pooling: Maintain persistent connections to avoid slow start.
Multiplexing and pipelining: Send multiple requests within a single connection to maximize network efficiency.
Adjust initial congestion window: Increase the initial packet allowance to expedite congestion window growth.
Use satellite-specific congestion control algorithms like BBR or CUBIC.

c) To improve bandwidth efficiency in the scenario of an online social network web client, a content-centric addressing scheme like Content-Centric Networking (CCN) or Named Data Networking (NDN) can be used. Instead of relying on server addresses, content is identified by unique names. When a user requests content, routers cache it, allowing subsequent requests within the village to be served locally, reducing reliance on remote servers and improving bandwidth efficiency. (Alternative: peer-to-peer)

3. (12 points) Kotlin.

(a) (3 points) The following code represents a suspend function that performs image download and updates the UI as needed. The line where the function is called is also listed. Rewrite the code, so that the function correctly updates the UI.

```
1  private val processingScope = CoroutineScope(Dispatchers.DEFAULT)
2
3  processingScope.launch{downloadPhoto(myUrl)}
4
5  private suspend fun downloadPhoto(imageUrl: String) {
6
7    withContext(Dispatchers.MAIN) {
8      downloadedImageView.setImageResource(R.drawable.processing);
9    }
10
11   val url = URL(imageUrl)
12   val bitmap =  BitmapFactory.decodeStream(url.openConnection().getInputStream())
13
14   downloadedImageView.setImageResource(bitmap)
15
16 }
```

(b) (3 points) The following function of MainActivity is called when accelerometer sensing is performed:

```
1  override fun onSensorChanged(event: SensorEvent?) {
2
3    val x = event.values[0]
4    val y = event.values[0]
5    val z = event.values[0]
6
7    textViewX.text = "X: $x"
8    textViewY.text = "Y: $y"
9    textViewZ.text = "Z: $z"
10 }
```

Explain the problem with the above code. Fix it using Kotlin's smart casting.

(c) (3 points) Refactor the following code using "apply":

```
1  val contentValues = ContentValues()
2  contentValues.put(MediaStore.MediaColumns.DISPLAY_NAME, "MyPage")
3  contentValues.put(MediaStore.MediaColumns.MIME_TYPE, "text/html")
4  contentValues.put(MediaStore.Downloads.RELATIVE_PATH, Environment.
       DIRECTORY_DOWNLOADS+File.separator+"Temp")
5  }
```

(d) (3 points) Translate the following Java code to Kotlin:

```
1  public class MainActivity extends AppCompatActivity {
2   public static final String TAG = "MainActivity";
3
4   @Override
5   public void onCreate(Bundle savedInstanceState) {
6    super.onCreate(savedInstanceState);
7    setContentView(R.layout.activity_main);
8   }
9  }
```

a) (pri vrstici 11-14):

```
withContext(Dispatchers.Main) {
  val url = URL(imageUrl)
  val bitmap = BitmapFactory.decodeStream(url.openConnection().getInputStream())
}
```

b)

```
override fun onSensorChanged(event: SensorEvent?) {
        val x = event?.values[0]
        val y = event?.values[1]
        val z = event?.values[2]

        textViewX.text = "X: $x"
        textViewY.text = "Y: $y"
        textViewZ.text = "Z: $z"
}
```

c)

```
val contentValues = ContentValues().apply {
    put(MediaStore.MediaColumns.DISPLAY_NAME, "MyPage")
    put(MediaStore.MediaColumns.MIME_TYPE, "text/html")
    put(MediaStore.Downloads.RELATIVE_PATH, "${Environment.DIRECTORY_DOWNLOADS}${File.separator}Temp")
}
```

4. (16 points) You are designing an Android real-estate ad browsing app. The app should get data about properties on the market from a remote REST server. The server sends data in the JSON format.

   (a) (4 points) UI: Describe how you would implement the app so that it has two views - one for showing a gallery view of the available real estate, and the other (opened once a user clicks on a selected property) showing the details of the property.

   (b) (4 points) Data manipulation: Describe the data transfer/manipulation/storage elements of your app. Clearly define the purpose of each of the elements you use.

   (c) (4 points) Data caching: To enable offline viewing, the app should provide a caching functionality. Describe how you would implement such a functionality.

   (d) (4 points) Synchronization: The data on the server can change, for example, because a new property appears, the price changes, or property gets sold (note: for now, assume that sold property still remains on the server, just with a field "sold" set to "true", i.e. it does not get deleted). Describe how you would ensure that the up-to-date data is fetched from the server once a day, if the phone is connected to a WiFi network and connected to a charger.

   Note: your solution should be as efficient as possible. Please draw diagram(s) of your solution.

a) 1 Activity with NavHostfragment and 2 Fragments EstatesListFragment & DetailsFragment
EstatesListFragment: RecycleViewer to store list of estates (e.g. CardView), uses Adapter to get data
DetailsFragment: Details regarding the estate (ImageView, textCiew,. . .)
The two are connected with nav graph, the ID is passed so the right estate details are shown

b) ItevViewModel which is stored in a LiveData object (observed by the Fragments). It communicates with the Repository class, which we use to access REST API and local database. For the local db, we

use Room and define functions with DAO, then we also need to define an @Entity object to have a table for data. For REST API, we use Retrofit and GSON, then define functions in DTO.
When the user first opens the app data from REST API is fetched and added in the local db. When offline, the app can now access that local db (of course we cannot buy the estates when offline).

c) mentioned in b)

d) With WorkManager and Repository we check for data changes every 24 hours and update them (posledično tudi Views) accordingly.