ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 6, No. 2, 2021

SENSORS IN CYBER-PHYSICAL SYSTEMS BASED ON ANDROID OPERATING SYSTEM

***Valerii Bielik*[1], *Yurii Morozov*[1], *Mykola Morozov*[2]**

[1] *Lviv Polytechnic National University, 12, S. Bandery Str., Lviv, 79013, Ukraine.*
[2] *Technical University of Munich, Boltzmannstr. 3, Garching b. Munich, 85748, Germany*
Authors' e-mail*: valerii.bielik.mki.2020@lpnu.ua, yurii.v.morozov@lpnu.ua*

https://doi.org/10.23939/acps2021.02.083

Submitted on 11.10.2021

***Abstract*: The cyber-physical systems take the major part of any system that help users to interact with environment processes.**

Cyber-physical systems are intelligent systems, which include networks of physical and computing components that interact on internal level. The basis for the development of various models of cyber-physical systems are the using of measuring instruments and their software. Measuring instruments are necessary to control technological parameters processes and the environment.

The purpose was to investigate the features of interaction with sensors, to identify the most useful of them in use, to classify types and describe their capabilities for future use in developing of cyber-physical systems.

The relevance of the choice of this topic is that mobile and cyber-physical systems occupy a significant place in modern life. The systems that help the user to simplify daily tasks are of maximum benefit. These tasks can be attributed to the tasks of the environment as they exist and are performed in it. Especially cyber-physical systems that interact with the environment have the ability to solve such problems. Sensors act as a tool of interaction, the so-called bridge between the environment and the program. Sensors collect and provide information for further processing and use in solving problems.

***Index Terms*: Android, camera, geolocation, image processing, Kotlin, MVVM, sensors.**

INTRODUCTION

Cyber-physical systems are intelligent systems and the goal of such intelligent systems is in use of sensors data that signal about the environment parameters change as quickly and accurately as possible, special algorithms that involve the higher-level automation to perform necessary actions. Usually cyber-physical systems cover all known aspects of operation with information and measuring systems, complicated by the interaction of their individual components through networks. They unite informational technologies: from getting data from sensors with their next processing using built-in computing power or using cloud technologies, to traditional operational control and management technologies. In other words, a feature of the cyber-physical systems is the combination of informational and operational technologies with some imposed time and space constraints [1].

The issue of introducing "intellectual" programs in various fields are so relevant that NIST has developed a cyber-physical systems classification that covers direct and additional intellectual production, smart structures, smart transport, smart energy, intellectual safety of life and smart health protection. To ensure the implementation of "intelligent" programs, it is necessary to create an application system model and have the appropriate sensors and software. In addition, it is necessary to achieve compatibility between dissimilar components and systems, so we need developments done in the field of metrology (calibration, quality assessment of complex products, model-based diagnostics), in the field of development of basic and intermediate software. The software creates adequate predicted behavior of the system – the system's response for changes of different parameters from sensors.

There are four types of cyber-physical systems:

Cyber-physical system "Smart Production" – multifunctional smart machines of small size, adaptive to user needs (implemented by assembling the required functionality on one machine). Having received information about changed requirements, cyber-physical system itself adjusts the technological process. An example of smart production is the production of metal using exact weights.

Cyber-physical system "Smart Buildings" – intelligent buildings (with minimal or zero resources consumption) that require constant monitoring. They must be connected to networks of intelligent sensors and monitored by technics of cyber-physical system. The main requirement in such systems is to achieve zero energy consumption.

Intermediate software is a prerequisite for effective work of cyber-physical system. It provides services for software applications, except those, that are available in the operating system, and connects software components and enterprise applications.

Smart measuring instruments are an important prerequisite for creating a cyber-physical system since they are the main information provided components and measuring subsystems. Smart measuring instruments are divided into the following subclasses: smart sensors, smart converters, their networks, which can be combined in modern wireless sensor networks.

The described cyber-physical system in this article is related to the third type. The intermediate software includes web servers, application servers, content management systems and similar tools that support application development and delivery and enable communication and data management in distributed applications. This is especially related to informational technology based on: advanced language markup, on the protocol of structured messaging in distributed computing systems when accessing objects (SOAP), on web services, on a modular approach (SOA) for software development that based on the use of distributed, weak related replacement components, equipped with standardized interfaces for interaction on standardized protocols, on infrastructure Web 2.0 and the protocol for easy access to directories (LDAP) [2].

Services that can be considered as intermediate programs include the integration of corporate applications, data integration focused on middleware messages, object request brokers (ORBs) and corporate service buses (ESBs).

Operational system Android can be another example of intermediate software. The Android operating system uses the Linux kernel and provides services that developers use for implementing into their programs. In addition, Android provides a level of intermediate software, including libraries, which provide services such as data storage, screen display, multimedia and browser.

In the article also were described sensors features of Android devices for software developers [3], [4]. Into account were taken motion, position and environmental sensors [5], [6]. Best practice for GPS using was described and analyzed with provided diagrams [7], [8]. General camera features were described. In addition, difference between old and new camera application programming interfaces were presented and highlighted the pros of the new one [9], [10].

GOAL

The main goal of the article is to investigate the features of Android device sensors, to describe their power in Android program developing, to show best ways of their implementation for developers. Also the additional goal is to develop a cyber-physical system, which will use camera, GPS and sensors of the Android mobile device to identify city objects. The system should be able to identify the object in 2 seconds with the recognition percentage equal about 90 %.

ANALYSIS OF RECENT RESEARCH AND PUBLICATIONS

Android device sensors are virtual devices that provide data from a set of physical sensors: accelerometers, gyroscopes, magnetometers, barometers, humidity, pressure, light, proximity and heart rate sensors. The camera, fingerprint sensor, microphone and touch screen are not included in the above list of physical devices as they have their own reporting mechanism. In addition, the sensors provide data with lower bandwidth. If to take into account the accelerometer, then it has a standard bandwidth of 100 Hz x 3 channels. For the camera, the bandwidth will be much higher – 25 Hz x 8 MP x 3 channels [3].

Android does not determine how different physical sensors are connecting to the system on a chip (SoC). Often, sensor chips are connected to the SoC via a touch hub, which allows some monitoring and data processing at low power. An inter-integrated circuit (I2C) or a serial peripheral interface (SPI) is used as the transport mechanism. To reduce power consumption, some architectures are hierarchical, with minimal processing performed in a special integrated circuit (ASIC – similar to detecting motion on the accelerometer chip), and the rest is done in a microcontroller (for example, detecting steps in the sensor hub).

Also important is the fact that each Android sensor has a “type” that indicates how the sensor behaves and what data it provides. The official types of Android sensors are defined in a special file that is publicly available. Sensors can be divided into two groups: formal and informal. Only officially approved sensors will be used to develop this system. The types of these sensors are documented in the Android SDK, a framework that allows you to develop applications for Android-based devices. Also, the behavior of sensors of this type is tested in a set of Android compatibility tests (CTS). In addition, official sensors will be available on Android-based mobile devices.

MOTION SENSORS IN ANDROID DEVICES

Motion sensors are needed to monitor the position of the device in space. The possible architecture of motion sensors depends on the type of sensor:

Sensors of gravity, linear acceleration, rotation vector, motion, step counter and step tracking sensors are either hardware or software.

Accelerometer and gyroscope sensors always have a hardware basis.

Most Android devices have an accelerometer, and now many of them include a gyroscope. The availability of software-based sensors is more diverse, as they often rely on one or more hardware sensors to obtain their data. Depending on the device, these software sensors can receive their data either from the accelerometer and magnetometer, or from the gyroscope.

Motion sensors are useful for monitoring the movement of the device, such as tilt, shake, rotation or swing. The motion is usually a reflection of the user's direct input (for example, the user driving the game or the user controlling the ball in the game), but it can also be a reflection of the physical environment in which the device is located (for example, moving with the user under driving time). In the first case, the movement relative to the reference system of the device or the reference system of the application is tracked; in the

second case, the movement relative to the world frame of reference is tracked.

All motion sensors return multidimensional arrays of sensor values for each measurement. For example, during a single sensor event, the accelerometer returns acceleration data for three coordinate axes, and the gyroscope returns speed data for three coordinate axes. These data values are returned in an array of floating-point values along with some other parameters.

Rotation vector sensors and gravity sensor are the most commonly used sensors for motion detection and control. For example, a vector rotation sensor is ideal if you are developing a game, an augmented reality application, a two-dimensional or three-dimensional compass, or an application to stabilize the camera. In most cases, the use of these sensors is a better choice than the use of an accelerometer and a geomagnetic field sensor or an orientation sensor.

The gravity sensor provides a three-dimensional vector that indicates the direction and magnitude of gravity. Typically, this sensor is used to determine the relative orientation of the device in space. The units are the same as for the acceleration sensor (m/s^2). The coordinate system is also the same as for the acceleration sensor. When the device is at rest, the gravity sensor provides data identical to the data collected by the accelerometer.

The linear acceleration sensor provides a three-dimensional vector representing the acceleration along each axis of the device, excluding the action of gravity. This value is usually used to detect gestures. The value can also serve as an input signal to the inertial navigation system, which uses the method of calculating coordinates.

This sensor is usually used when it is necessary to obtain data on acceleration without the influence of gravity. For example, you can use this sensor to see how fast the device is moving. The linear acceleration sensor always has an offset that needs to be removed. The easiest way to do this is to incorporate a calibration step into your program. During calibration, you can ask the user to place the device on a table and then read the offset for all three axes. It is then necessary to subtract this offset from the direct readings of the acceleration sensor to obtain the actual linear acceleration. The coordinate system of the sensor is the same as for the acceleration sensor, as well as the unit of measurement (m/s^2).

The rotation vector sensor represents the orientation of the device as a combination of the angle and the axis at which the device rotated at an angle θ around the axis (x, y or z). These three elements of the rotation vector are expressed as follows: $x \cdot \sin(\theta/2)$, $y \cdot \sin(\theta/2)$, $z \cdot \sin(\theta/2)$. Where the magnitude of the rotation vector is equal to $\sin(\theta/2)$, and the direction of the rotation vector is equal to the direction of the axis of rotation.

POSITION SENSORS

The Android platform offers two sensors that allow you to determine the position of the device: a geomagnetic field sensor and an accelerometer [4]. The Android platform also provides a sensor that allows you to determine how close the front of the device is to an object (known as a proximity sensor). The geomagnetic field sensor and the proximity sensor have a hardware basis. Most phone and tablet manufacturers have a geomagnetic field sensor. Similarly, manufacturers of devices that allow you to make calls usually include a proximity sensor to determine when the device is being held close to the user's face (for example, during a phone call). To determine the orientation of the device, you can use the readings of the device accelerometer and geomagnetic field sensor.

The geomagnetic rotation vector sensor is similar to the rotation vector sensor, but does not use a gyroscope. The accuracy of this sensor is lower than that of a conventional vector rotation sensor, but the power consumption is reduced. It is better to use this sensor only if you need to collect information about the rotation in the background without consuming too much battery power. This sensor is most useful when used in conjunction with filtering and additional calculations. By calculating the orientation of the device, it will be possible to track the position of the device relative to the Earth's reference system.

To solve this problem, the system must calculate the orientation angles using the geomagnetic field sensor of the device in combination with the accelerometer of the device. Using these two hardware sensors, the system provides data for the following three orientation angles:

Azimuth (degrees of rotation around the z axis). This is the angle between the current compass direction of the device and north. If the upper edge of the device faces north, the azimuth is 0 degrees; if the upper edge faces south, the azimuth is 180 degrees. Similarly, if the upper edge faces east, the azimuth is 90 degrees, and if the upper edge faces west, the azimuth is 270 degrees.

Tilt (degrees of rotation around the x-axis). This is the angle between the plane parallel to the device screen and the plane parallel to the ground. If you hold the device parallel to the ground with the lowest edge closest to the user and tilt the top edge of the device to the ground, the tilt angle becomes positive. Tilting in the opposite direction – moving the upper edge of the device off the ground – causes the tilt angle to become negative. Value range from -180 to 180 degrees.

Rotation (degrees of rotation around the y-axis). This is the angle between the plane perpendicular to the screen of the device and the plane perpendicular to the ground. If you hold the device parallel to the ground with the lowest edge closest to the user and tilt the left edge of the device to the ground, the tilt angle becomes positive. Tilting in the opposite direction – moving the right edge of the device to the ground – causes a negative tilt angle. Values range from -90 degrees to 90 degrees.

It is also important to note that these angles work with a different coordinate system than the one used in aviation.

The geomagnetic field sensor allows you to track changes in the Earth’s magnetic field. This sensor

provides initial data on the field strength (in μT) for each of the three coordinate axes. Normally, you do not need to use this sensor directly. Instead, you can use a vector rotation sensor to detect raw rotational motion, or, alternatively, use an accelerometer and a geomagnetic field sensor together with the getRotationMatrix() method to obtain a rotation matrix and a tilt matrix. Then use these matrices with the getOrientation() and getInclination() methods to obtain azimuth and geomagnetic data [5].

An uncalibrated magnetometer is similar to a geomagnetic field sensor, except that no solid iron calibration is applied to the magnetic field. Factory calibration and temperature compensation are still applied to the magnetic field. An uncalibrated magnetometer is useful for handling poor geomagnetic field sensor ratings. Uncalibrated sensors give more raw results and may include some biases, but their measurements contain fewer jumps from the corrections applied by calibration. Some programs may prefer these calibrated results because they are smoother and more reliable.

ENVIRONMENTAL SENSORS

The Android platform offers four sensors that allow to monitor various properties of the environment. These sensors can be used to monitor relative humidity, light, ambient pressure, and ambient temperature near an Android device. All four environmental sensors are hardware-based and only available if the device manufacturer has built them into the device. With the exception of the light sensor, which most device manufacturers use to monitor screen brightness, environmental sensors are not always available on devices. With this in mind, it is especially important to check at runtime for an environment sensor before attempting to retrieve data from it [6].

GEOLOCATION

Proper use of location information can be beneficial for users. For example, if a program helps a user navigate while walking or driving, or if it tracks the location of objects, they need to get the location of the device regularly [7]. In addition to the geographical location (latitude and longitude), it is possible to provide the user with additional information, such as landmark (horizontal direction of movement), height or speed of the device. This information and more is available in the Location object, which the application can obtain from the device location provider. In response, the API periodically provides the program with the best available location based on existing location providers, such as WiFi and GPS (Global Positioning System). The providers, the location permissions that have been set and the parameters that are set directly in the location request determine location accuracy [8].

Android devices have several hardware sensors that developer can use to calculate the location of the device.

The GPS module is a special location module that uses satellite signals to determine location, so it can only be used outdoors. It is very accurate, but also consumes a lot of energy.

WiFi module – developers can use the WiFi-RTT API, which is available in Android 9 and above, to measure the distance to the nearest WiFi access points. Then it is possible to use these distances to determine the location of the device with an accuracy of 1–2 meters. It does not consume too much power as the GPS module does.

Sensors – most modern Android phones are equipped with sensors that monitor the movement of the device, and they also fill the gap, which will be described below.

Let use the diagram to illustrate how these hardware modules and sensors assess the location of the device.

The x-axis shows the time, and the y-axis shows the accuracy of the location. In this example, the user is first in the open air, enters the indoor environment, and then goes outside again. If only the GPS module is used to estimate the location, then Fig. 1 shows the diagram we get.

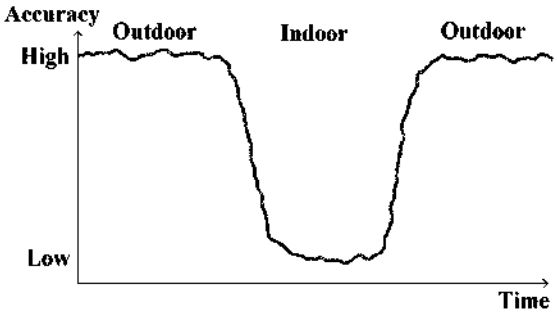


Fig. 1. The scheme of accuracy measurement of a location by means of the GPS module

In an outdoor environment, GPS works perfectly because the device is capable of receiving satellite signals. But when it comes to the indoor environment, the satellite signal is lost and the GPS module is unable to provide the data needed to determine the location.

Fig. 2 shows what will happen if the WiFi module is used.

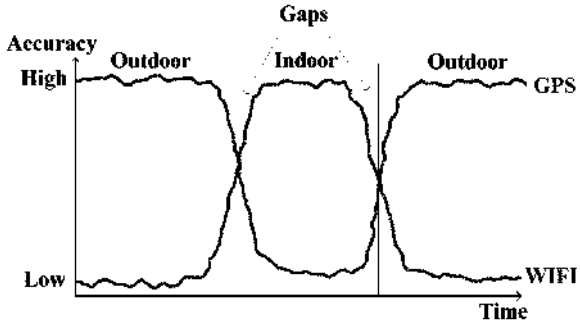


Fig. 2. The scheme of accuracy measurement of a location by means of the WIFI module

This is the exact opposite of what was obtained on the chart only from GPS. In a room where WiFi access points are available, we get great accuracy for location determining. When users are outdoors and WiFi module is used, but where no WiFi access points then no signals are received, so it is not possible to determine the location in this case. So the charts show that GPS works great outdoors, and WiFi works great indoors.

By combining signals from the GPS module and the WiFi module, it is possible to get a fairly good location estimate, which covers the scenarios of the user's location inside and outside.

However, there are two visible gaps in this diagram. Especially when the device goes from external to internal environment, or vice versa. GPS and WiFi signals are not available during these times.

The combination of gyroscope, accelerometer, and magnetometer signals can help fill in the gaps using a Bayesian synthesis algorithm. As a result, we obtain the scheme shown in Fig. 3.

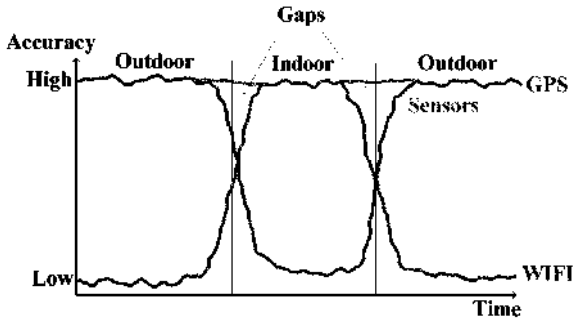


Fig. 3. Diagram of combining the results of location measurement accuracy using GPS modules, WiFi and sensors

Summing up all together:

The GPS module is used to determine the location outdoors.

The WiFi module is used to determine the location in the room.

Sensors are used to fill in the gaps between transitions between the use of different modules.

So we have a good location estimate that covers all scenarios. But does the system need to use an API to directly collect data from these sensors? There are LocationManager and SensorManager to help interact with the GPS module and other sensors, and there is a WiFi location API that can help to get WiFi-RTT. But applying the Bayesian Fusion algorithm to combine these signals is a challenge.

To do this, there is an API Fused Location Provider, which is part of the Google Library. It cleverly combines different signals to calculate the location information your application needs. It provides a powerful, high-level structure by automating location provider selection and power management. It uses FusedLocationProviderApi to automatically select a base provider based on accuracy, battery usage, and more. Before requesting a location update, the system must connect to location services and send a location request.

If compare the API described above with what Android sdk offers, then both of these approaches are good to use when getting location results. But in the official Android documentation, it is strongly recommended to use Google location services, rather than the standard Android sdk API, due to the fact that it automates the choice of location provider and power management.

Google's location services offer significant advantages over the Android Framework Location API. This gives faster results because the results are obtained from a system-wide service that constantly updates it. There is also more stability during use and lower battery consumption. It is also possible to use advanced features such as geozoning.

The only disadvantage is that Google location services require Google Play services to be installed on the device to use location provider.

CAMERA

The Android framework includes support for various cameras and camera features available on devices, allowing you to take photos and videos of the developers. Android sdk provides two libraries for interacting with the camera: Camera and CameraX. The first is outdated, so it is impractical to use. CameraX, in turn, provides a wider range of opportunities for developers [9].

The developers use CameraX to interact with the device's camera using an abstraction called a use case. The following usage options are currently available:

Preview: Creates a surface for displaying a preview, such as PreviewView.

Image Analysis: Provides processor buffers for analysis, such as machine learning.

Image capture: Captures and saves a photo.

Usage scenarios can be combined and activated at the same time. For example, the app can allow a user to view an image seen by the camera using a preview option, have an image analysis usage scenario that determines if people are smiling in the photo, and include a script to use the image to take pictures when they are.

To work with the library it is necessary to specify the following:

Preferred use with configuration parameters.

What to do with the source data.

Predict a scenario, such as when to turn on cameras and when to store data, by linking usage options to Android lifecycle lifestyles.

You must configure usage options using the set() methods and complete them with the build() method. Each usage script object provides a set of APIs for specific uses. For example, an image capture script provides a call to the takePicture() method.

Instead of the system placing specific start and stop method calls in onResume() and onPause(), the program specifies the life cycle to which the camera should be connected using cameraProvider.bindToLifecycle(). This

life cycle then notifies CameraX when the camera capture session needs to be configured, and provides a corresponding change in the state of the camera according to the transitions in the life cycle. CameraX tracks the life cycle to determine when to open the camera, when to create a shooting session, and when to stop and shut down. Usage script APIs provide method calls and callbacks to monitor progress. It is possible to link different usage scenarios to a single life cycle. If the system needs to support usage scenarios that cannot be combined, you must do one of the following:

Group compatible use cases together into more than one snippet, and then switch between snippets.

Create your own lifecycle component and use it to manually control the camera lifecycle.

In the cases described above, it is nessesary to make sure that not all usage scripts are associated with CameraX using ProcessCameraProvider.unbindAll() or by disconnecting each usage script separately. In addition, when these usage scenarios are tied to the lifecycle, it is possible to allow CameraX to control the opening and closing of the shooting session and the management of the system lifecycle [10].

SYSTEM OVERVIEW

This research was done to determine the developing process of the cyber-physical system for city objects identification with confirmation. This system requires use of camera, for getting camera preview video stream capture. Then, the video stream should be divided into frames and each frame should be processed with necessary logic. This logic should help to remove noises from the frame, to facilitate the image content of the frame and to transform it to the monochrome image. After the frames are processed, the system should be able to read the number of the building, captured by the camera. For this, it is necessary to use optical symbol recognition library. The library should also provide some trained data to achieve good results in symbols recognition. In parallel, in background, the system should capture the devices geolocation and data from sensors, such as accelerometer with geomagnetic sensor to determine the position of devise for the confirmation. Also the device position can be fetched to explore the city object features with the help of the google services.

The scheme, shown on Fig. 4. represents the work of the system from Android operating-based system perspective. All the time-consumption logic is executing in the background, with the purpose not to load the main execution thread. Data updates from sensors are collecting in an asynchronous way, with the specified time delay, with the purpose to retrieve the latest and the most precious data. Also all results are collected and proceeded in background, but they are moved to foreground to be displayed to the user.

The system was tested on the Android device with the camera characteristics: 48MP+8MP+2MP, aperture f/1.79, resolution 3840x2160 pixels, sensor size 5.06 mm × 3.79 mm, focal length 3.81 mm. The testing showed that 3 seconds delay for collecting data, provided by sensors, is enough to get precise results of user device location calculation.

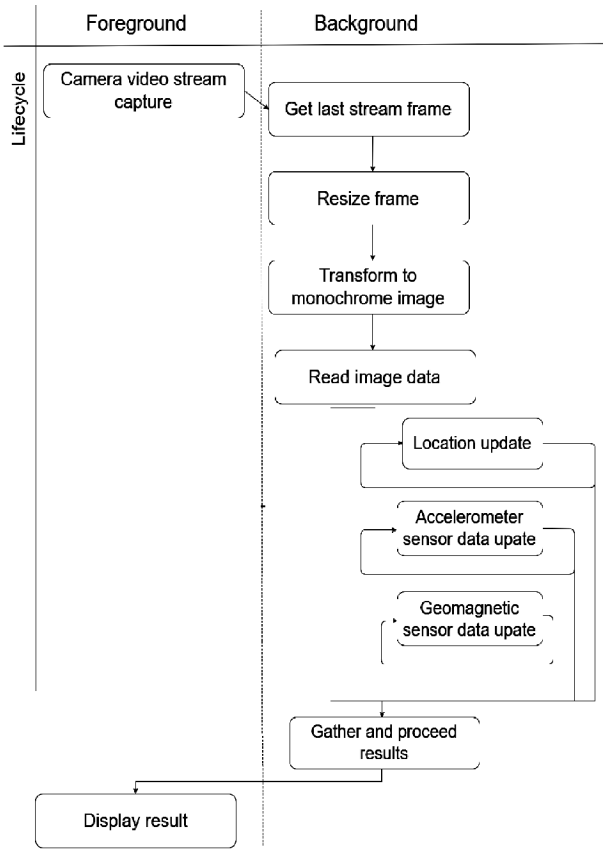


Fig. 4. The scheme of the cyber-physical system for city objects identification with confirmation

CONCLUSION

Features of using sensors of devices with Android operating system are considered and described. Motion, position and environmental sensors of Android operating-based system were analyzed. Their capabilities were described and possible scenarios for their use were provided. These scenarios were compared and the best solutions were suggested. Fused Location Provider API was considered as the best solution for geolocation services using in Android operating-based systems. Also the diagrams were provided to display the advantages of this geolocation service. Different modern frameworks for various camera use cases were described. CameraX framework was considered as the best solution for solving camera-related tasks on Android operating-based systems.

The developed system was tested on device with such camera default characteristics: 48MP+8MP+2MP, aperture f/1.79, resolution 3840×2160 pixels, sensor size 5.06 mm × 3.79 mm, focal length 3.81 mm. Also, accelerometer, geomagnetic field sensor and gps were used in the system. The sensors parameters data of Android device are hidden by manufacture. Accelerometers' standard bandwidth is 100 Hz × 3 channels. For the camera, the bandwidth is much higher – 25 Hz x

8 MP × 3 channels The sensors data was collected every 3 seconds with the purpose not to overload the system and at the same time receive precise results.

REFERENCES

[1] Pasternak I. and Morozov Y., (2014). Computational Problems of Electrical Engineering. *Modular network interface for distributed information navigation systems*. Lviv, pp. 15–25. Available at: http://ena.lp.edu.ua:8080/handle/ntb/33711 (Accessed: 9 October 2021)

[2] Gotra Z., Goliaka R., Morozov Y. and Halavka A., (1995). Bulletin of the Lviv Polytechnic State University. *Standby mode of operation of bipolar sensor ICs of multichannel information and measuring systems.* Lviv, pp. 131–136.

[3] Kyoung-Dae Kim and Kumar P. R., (2012). Wireless Sensor Networks. *Cyber–Physical Systems: A Perspective at the Centennial Proceedings of the IEEE*. Italy: IEEE. pp. 1287–1308. DOI: 10.1109/JPROC.2012.2189792

[4] Jiafu W., Hehua Y., Hui S. and Fang L., (2011). KSII Transactions on Internet and Information Systems. *Advances in Cyber-Physical Systems Research.* China. pp. 1891–1908. DOI:10.3837/tiis.2011.11.001

[5] Castano F., Strzelczak S., Villalonga A., Rodolfo E. Haber and Kossakowska J., (2019). Reliability of Sensors in Cyber-Physicals Systems. A Brief State-of-the-Art Review. *Sensor Reliability in Cyber-Physical Systems Using Internet-of-Things Data: A Review and Case Study*, Spain, pp. 3–12. DOI:10.3390/rs11192252

[6] Pradeepkumar Ashok, Ganesh Krishnamoorthy and Delbert Tesar., (2011). Cyber physical systems. *Guidelines for Managing Sensors in Cyber Physical Systems with Multiple Sensors*. USA: Journal of Sensors. pp. 72–84. DOI:10.1155/2011/321709

[7] Meike G. and Schiefer L., (2021). Advantages and tradeoffs using Android in smart IoT devices. *Inside the Android OS: Building, Customizing, Managing and Operating Android System Services (Android Deep Dive)*. USA. pp. 160–272. ISBN-13: 9780134096346

[8] Yaghmour K., (2013). Android's non-recursive build system. *Embedded Android Porting, Extending, and Customizing*. USA. pp. 45–56. ISBN: 9781449308292

[9] Haseman C., (2008). How location-based services are becoming more and more important in the mobile world. *Android Essentials*. USA. pp. 75–87. ISBN-13: 9781430210634

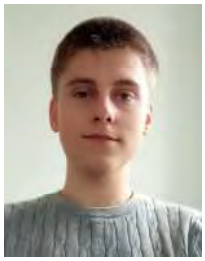
[10] Gargenta M., (2011). Overview of the Android platform and discover how it fits into the mobile ecosystem. *Learning Android*. USA. pp. 76–90. ISBN: 9781449390501

Valerii Bielk is a sixth-year computer engineering student of Lviv Polytechnic National University. Since 2019, he has been working in Pecode.

His research interests include data processing, Android developing, data-oriented programming, artificial intelligence systems, cyber-physical systems.

Yurii Morozov is a candidate of Technical Sciences, associate professor of the department of electronic computing machines, Institute of computer technology, automation and metrology, Lviv Polytechnic National University.

His research interests: creation of systems of complex information protection (design of virtual communication networks (VPN), instruments for information codding, systems of delimitation of access to information, instruments of the analysis of stability of networks, mechanisms of detection of attacks).



Mykola Morozov was born in 2000 in Lviv, Ukraine. He received the B.S. degree in software department at Lviv Polytechnic National University in 2021. He has been doing scientific and research work since 2017. Currently, he is a Postgraduate student of Informatics Department at Technical University of Munich. His research interests include architecture in cyber-physical systems and pathfinding algorithms.