به نام خدا

آشنایی با شبکه های تلفن های همراه
پروژه باریم (Barium)

بکتاش انصاری ۹۹۵۲۱۰۸۲
سیده شکیبا انارکی فیروز ۹۹۴۴۲۰۴۷

بهار ۱۴۰۳

عنوان پروژه: پروژه باریم (Barium)

شرح موضوع:

گسترش شبکه های تلفن همراه، به ویژه شبکه های نسل چهار و پنج، آنها را به بزرگترین شبکه دسترسی برای اینترنت تبدیل کرده است. امنیت برنامه های کاربردی و ارتباط امن، یکی از چالش های اصلی این حوزه است. امنیت در ارتباط از طریق شبکه های تلفن همراه، مسئله ای پیچیده است که باید به آن پرداخته شود. در پروژه فرض شده است که یک برنامه کاربردی از طریق پیامک (SMS) با یک خدمتگزار ارتباط برقرار می‌کند، که باید به نکات امنیتی و اجرایی آن توجه شود. این ارتباط باید مستند شده و بر پایه پروتکل SMPP باشد.

برای سهولت درک، در اینجا فرض کنید که دو گوشی در اختیار داریم: یک گوشی مشتری و یک گوشی که ما به عنوان خدمتگزار استفاده می‌کنیم. در گوشی خدمتگزار، یک برنامه Android با عملکرد Backend نصب می‌شود. مشتری از طریق پیامک، دستورات را به خدمتگزار ارسال می‌کند. مشتری باید به صورت مداوم اطلاعات مربوط به توان دریافتی، تکنولوژی سلول خدمتگزار و مکان دریافت این اطلاعات را در صورت کاهش توان زیر یک سطح آستانه معین، به صورت یک پیام به خدمتگزار ارسال کند. همچنین زمانی که مشتری درخواست خود را برای خدمتگزار ارسال می‌کند، خدمتگزار باید درخواست را اجرا کرده و پاسخ را در یک پیامک جداگانه به مشتری ارسال کند.

- لازم به ذکر است که تمامی مراحل این پروژه به صورت گروهی پیاده سازی و آزمایش شده است . کد های مربوط به بخش سرور با سیستم آقای انصاری و کد های مربوط به کلاینت با سیستم خانم انارکی به GitHub repository اضافه شده است.

اجزا پیاده سازی شده:

سرور SMPP

در ابتدا، ما یک کد برای سرور SMPP پیاده سازی کردیم تا بتواند وظایف زیر را انجام دهد:

- به عنوان یک سرور، به مودم SMPP متصل می‌شود:
- این بخش از کد، ارتباط سرور با مودم SMPP را برقرار می‌کند تا بتواند پیام ها را دریافت و ارسال کند.

- استفاده از تابع هش (Hashing):
- از تابع هش برای تبدیل داده ها به مقدار هش شده استفاده می‌شود تا امنیت داده ها افزایش یابد.

- رمزنگاری (Encryption):
- این بخش از کد، داده ها را با استفاده از الگوریتم های رمزنگاری AES، رمزنگاری می‌کند تا اطلاعات محرمانه محافظت شوند.
- رمزگشایی (Decryption):
- برای خواندن و بازگشایی داده های رمزنگاری شده، از روش های رمزگشایی مناسب استفاده می‌شود.

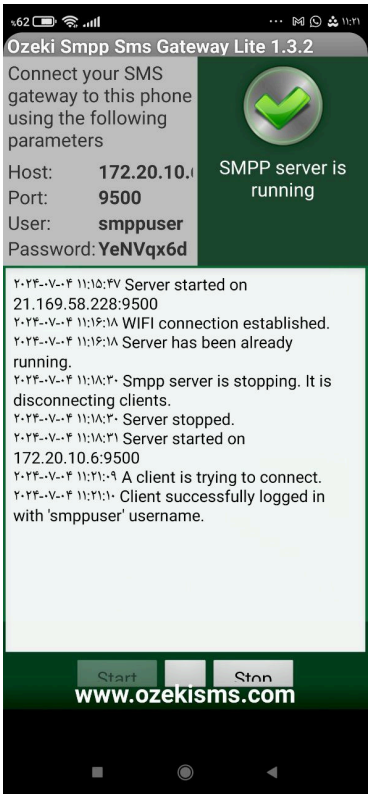
- ایجاد جدول داده در پایگاه داده:
- برای سازماندهی و ذخیره دادهها به صورت منظم، جداول داده در پایگاه داده ایجاد می‌شوند تا امکان جستجو و مدیریت داده ها بهبود یابد.
- ذخیره در پایگاه داده:
- داده های پردازش شده و امنیتی پس از هش، رمزنگاری و رمزگشایی، در پایگاه داده ذخیره می‌شوند تا بتوانند برای موارد آینده بهره برداری شوند.

- ارسال و دریافت پیام‌های متنی (SMS):

همان طور که پیش تر ذکر شد، سرور باید همیشه آماده دریافت پیام از سوی کاربران باشد تا بتواند وضعیت خود را به‌روزرسانی کرده و پیام تأیید (ack) را برای آنان ارسال کند. به همین منظور، سرور در یک حلقه `while true` دائماً منتظر دریافت پیام از سوی کاربران (برنامه گوشی) است، مگر اینکه کاربر سرور فرآیند را با فشردن کلیدی متوقف کند. در صورت بروز هرگونه خطا، سرور به طور پیش‌فرض تلاش می‌کند تا ارتباط را مجدداً برقرار کرده و به انتظار برای دریافت پیام ادامه دهد.

SMPP router

ما یک موبایل را به عنوان یک مودم GSM با استفاده از پروتکل SMPP در نظر می‌گیریم. برای این منظور، نیاز است که گوشی خود را به یک مودم SMPP برای سرور تبدیل کنیم. برای این کار، می توان از برنامه های شبیه سازی گوشی استفاده کرد که در این پروژه از برنامه Ozeki Smpp Sms Gateway Lite استفاده می‌شود.
عکس زیر نشان دهنده اتصال برنامه اوزکی به سرور ما است که بر روی یک کامپیوتر در حال اجرا می‌باشد.



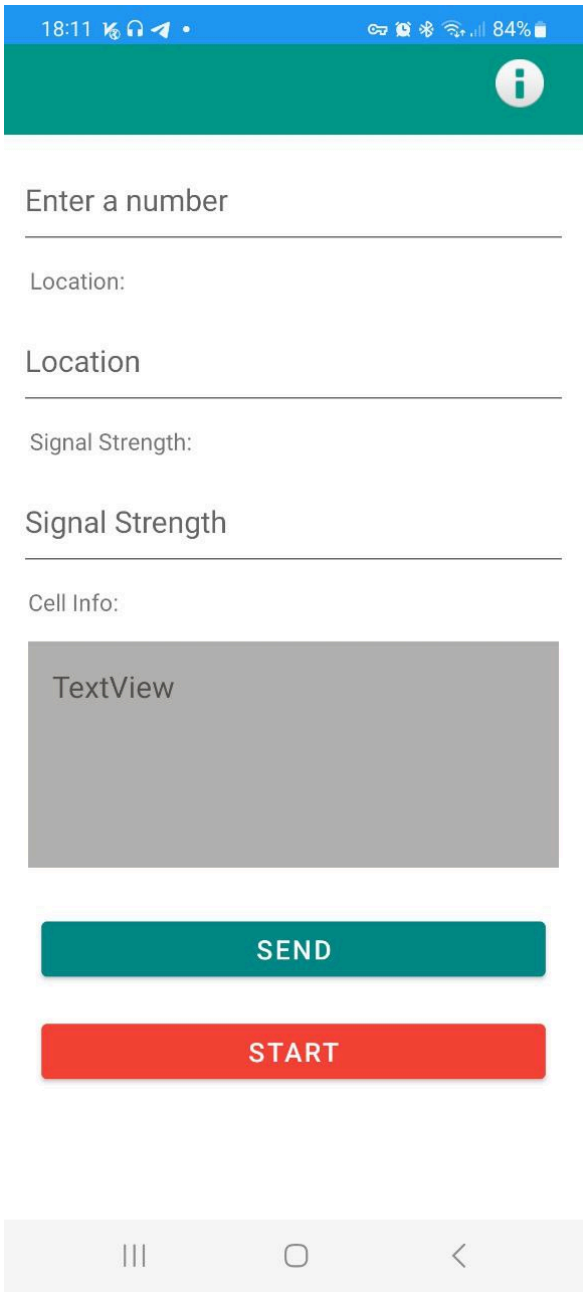
برنامه اندروید client

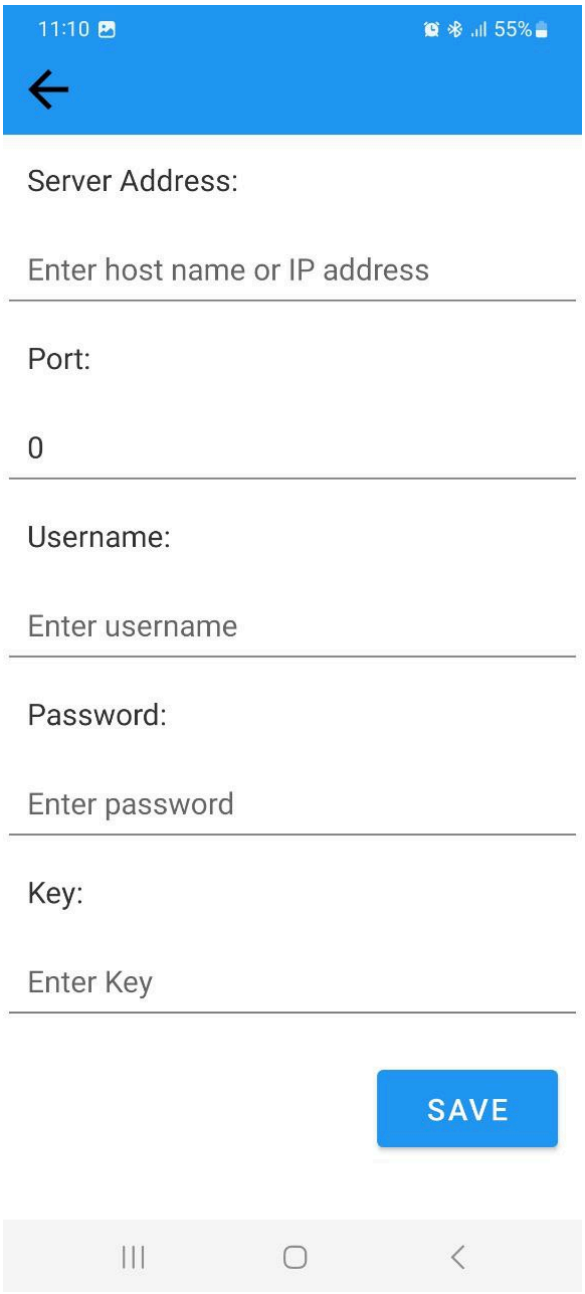
این برنامه برای اجرا بر روی تلفن همراه کلاینت با زبان کاتلین طراحی شده است و دارای قابلیت های زیر می‌باشد:

- اتصال به روتر SMPP
- اندازه گیری قدرت سیگنال تلفن همراه در شبکه های نسل های مختلف
- استخراج اطلاعات سلول خدمت رسان (serving cell) و سلول های همسایه
- جمع آوری داده های location
- به روزرسانی مستمر اطلاعات جمع آوری‌شده در بازه های زمانی ۵ ثانیه به طور مکرر
- ارسال اطلاعات سلول ها به سرور به صورت خودکار هرگاه که قدرت سیگنال به کمتر از مقیاس ۲.۵ برسد.
- امکان ارسال اطلاعات location و serving cell و سلول های همسایه به سرور توسط کاربر هم محقق شده است.

این برنامه قادر است به صورت خودکار اتصال به روتر SMPP را برقرار کند و با استفاده از داده های موقعیتی و اطلاعات سلول های همسایه آنها را به سرور ارسال کند تا امکان تحلیل و بهبود کیفیت شبکه را فراهم سازند. به علاوه، برنامه به کاربر این امکان را می دهد تا وضعیت سیگنال را به صورت زنده مشاهده کرده و در صورت نیاز اقدامات لازم را برای بهبود آن انجام دهد.

در ادامه تصاویر محیط برنامه را مشاهده می کنید. در صفحه ی اول اطلاعات نمایش داده می شوند و در صفحه ی دوم تنظیمات مربوط به اتصال هستند.





بخش هایی از کد پیاده سازی نرم افزار client اندروید :

```
// Status flags
private var isSMSSent = false
private val smsPermission: String = Manifest.permission.RECEIVE_SMS
private val smsRequestCode: Int = 2

// Location and telephony managers
private lateinit var locationHandler: Handler
private lateinit var locationRunnable: Runnable
// Threshold for signal strength
private var signalStrengthThreshold = 2.5
private lateinit var locationManager: LocationManager
private lateinit var telephonyManager: TelephonyManager

private lateinit var fusedLocationClient: FusedLocationProviderClient

private var isMonitoringActive = false
private val locationUpdateInterval = 5000 // 5 seconds

shakiba anaraki firooz
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    supportActionBar?.hide()

    val numberEditText = findViewById<EditText>(R.id.number_input)
    val locationEditText = findViewById<EditText>(R.id.loc_input)
    val signalStrengthEditText = findViewById<EditText>(R.id.value_input)
    val cellInfoTextView: TextView = findViewById(R.id.cell_info_input)
```

تنظیم پارامترهای اولیه: شامل آستانه قدرت سیگنال و بازه زمانی حلقه ی while.

```
settingsButton.setOnClickListener { it: View!
    val intent = Intent( packageContext: this, SettingsActi
    startActivity(intent)
}

com.example.smpp_shak_bak.R.id
public static abstract final int settings_button =
SMPP_SHAK_BAK.app.main

// Load encryption key from shared preferences
val sharedPref = this.getSharedPreferences( name: "gateway_config", MODE_PRIVATE)
val encryptionKey = sharedPref.getString("key", "")

// Send button click listener
val sendButton = findViewById<Button>(R.id.send_button)
sendButton.setOnClickListener { it: View!
    displaySendStatus( isSuccess: false,  isFailure: false)
    displayAckStatus( isSuccess: false,  isFailure: false)
    recipientNumber = numberEditText.text.toString()
    val location = locationEditText.text.toString()
    val value = signalStrengthEditText.text.toString()
    val cell = cellInfoTextView.text.toString()

    messageText = "LOC $location \n VAL $value \n CINFO $cell"
    isSendPending = true
}
```

عملکرد دکمه ارسال و فرآیند تنظیم داده ها

```
    // Define PhoneStateListener to get signal strength updates
    private val signalStrengthListener = object : PhoneStateListener() {
        override fun onSignalStrengthsChanged(signalStrength: SignalStrength) {
            super.onSignalStrengthsChanged(signalStrength)
            val signalLevel = signalStrength.level
            val signalStrengthEditText = findViewById<EditText>(R.id.value_input)
            signalStrengthEditText.setText(signalLevel.toString())

            if (signalLevel < signalStrengthThreshold) {
                Toast.makeText( context: this@MainActivity,  text: "Sending to server", Toast.LENGTH_SHORT).show()
                // Click on Send button to notify server
                val sendButton = findViewById<Button>(R.id.send_button)
                sendButton.performClick()
            }
        }
    }
}
```

شنونده قدرت سیگنال برای بررسی تغییرات

```
shakiba anaraki firooz
override fun onLocationChanged(location: Location) {
    val locationEditText = findViewById<EditText>(R.id.loc_input)
    locationEditText.setText("Lat: ${location.latitude}, Lon: ${location.longitude}")
}

shakiba anaraki firooz
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == this.smsRequestCode && grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        startMonitoring()
    }
}


shakiba anaraki firooz
private fun displayAckStatus(isSuccess: Boolean, isFailure: Boolean) {
    val successAckTextView = findViewById<TextView>(R.id.status_sa_textview)
    val failureAckTextView = findViewById<TextView>(R.id.status_fa_textview)

    successAckTextView.visibility = if (isSuccess) View.VISIBLE else View.INVISIBLE
    failureAckTextView.visibility = if (isFailure) View.VISIBLE else View.INVISIBLE
}

shakiba anaraki firooz
private fun displaySendStatus(isSuccess: Boolean, isFailure: Boolean) {
    val successSendTextView = findViewById<TextView>(R.id.status_sm_textview)
    val failureSendTextView = findViewById<TextView>(R.id.status_fm_textview)
```

توابع کمکی برای مدیریت حالت های مختلف

روند انجام کار:

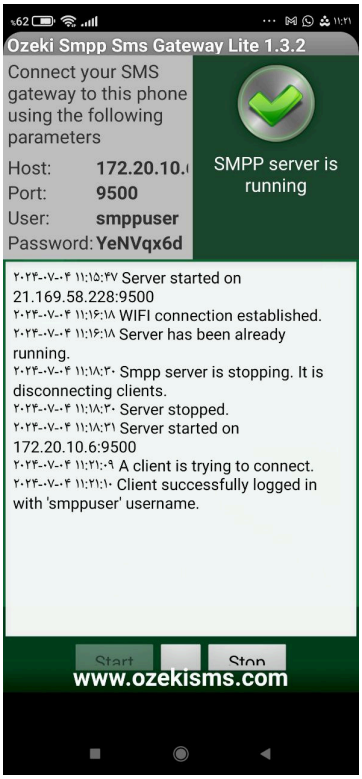
اجرای سرور SMPP:

ابتدا کد سرور SMPP را بر روی کامپیوتر اجرا می‌کنیم و اطمینان حاصل می‌کنیم که سرور آماده به کار است.

اتصال موبایل روتر به سرور:

سپس موبایل روتر را به سرور متصل می‌کنیم در حالی که هر دو دستگاه در یک شبکه مشترک (مانند شبکه محلی یا wifi) قرار دارند.

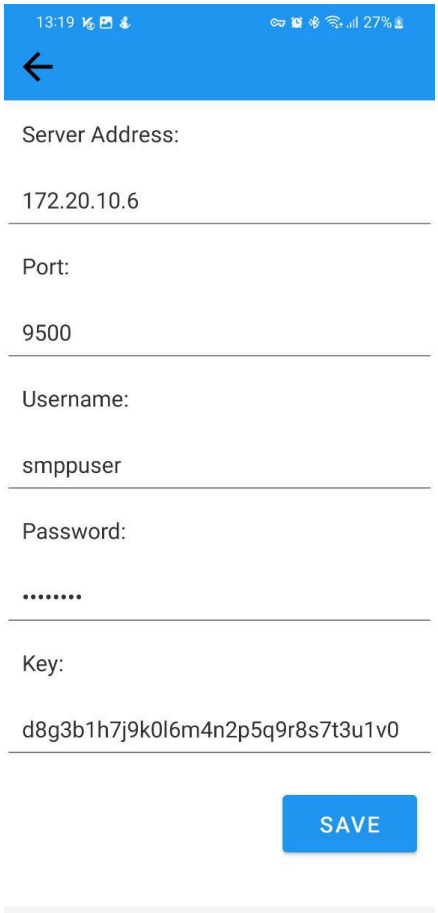
همچنین لازم است که اطلاعات اتصال برنامه با سرور همسان سازی شود.




```
# Initialize and start the SMPP message handler
smpp_handler = SMPPMessageHandler()
host = '172.20.10.6'
port = 9500
username = "smppuser"
password = "YeNVqx6d"
src_addr = '9377426044'
smpp_handler.send_and_receive_sms(host, port, username, password, src_addr)
```

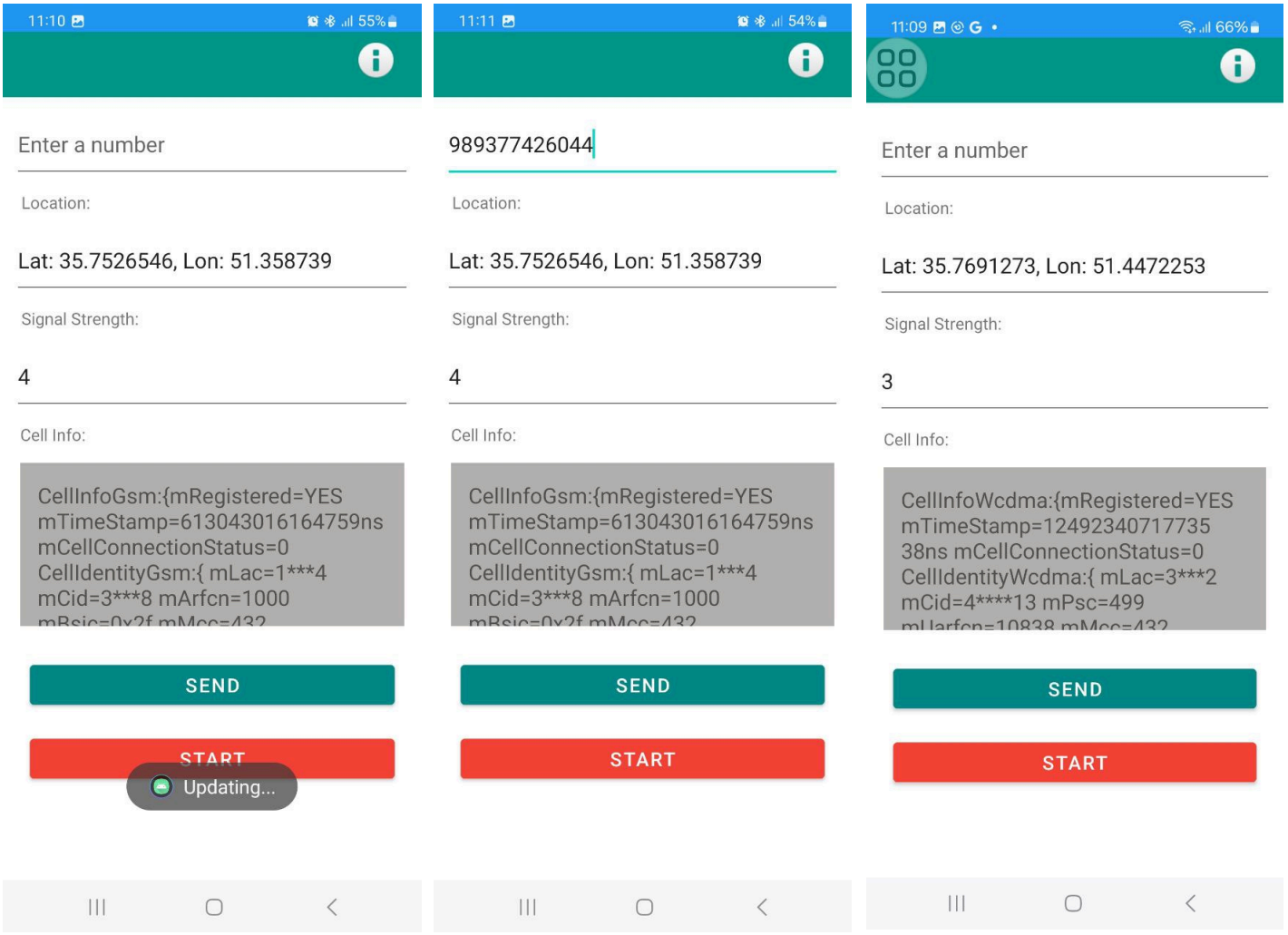
وارد کردن اطلاعات روتر در برنامه اندروید کلاینت:

بعد از آن، داده های روتر را در برنامه اندروید کلاینت وارد می‌کنیم، که شامل تنظیمات لازم برای ارتباط صحیح با آن می‌باشد.

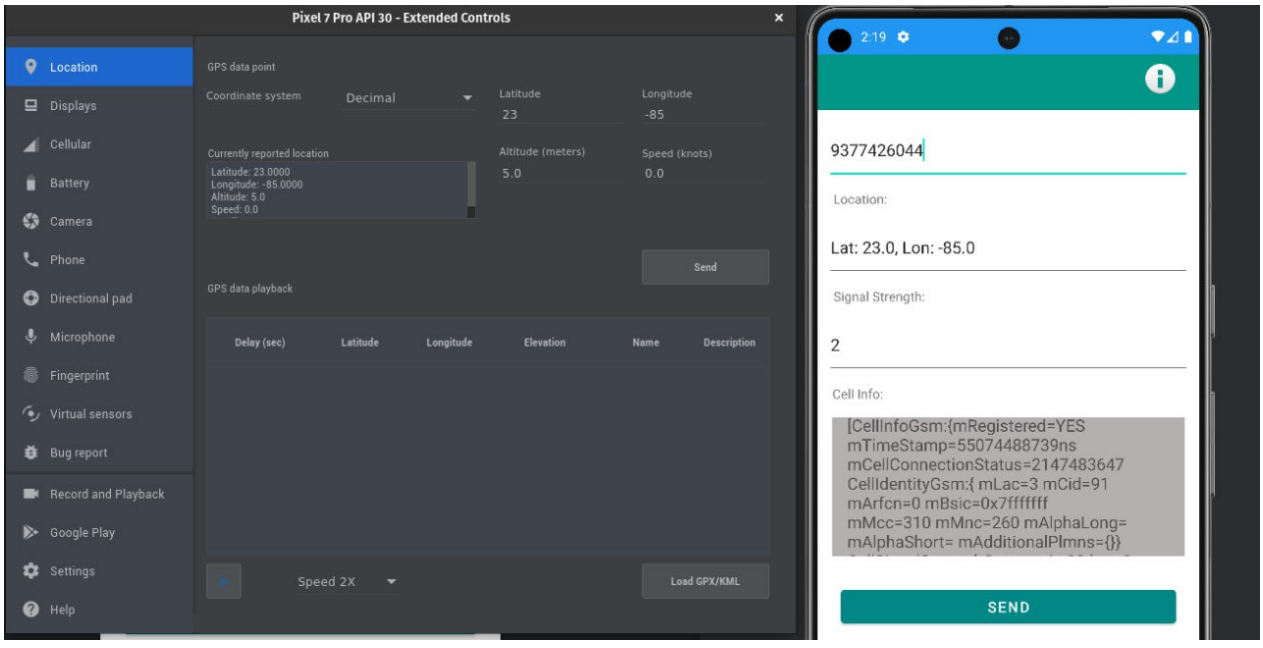


آغاز اندازه گیری سیگنال:

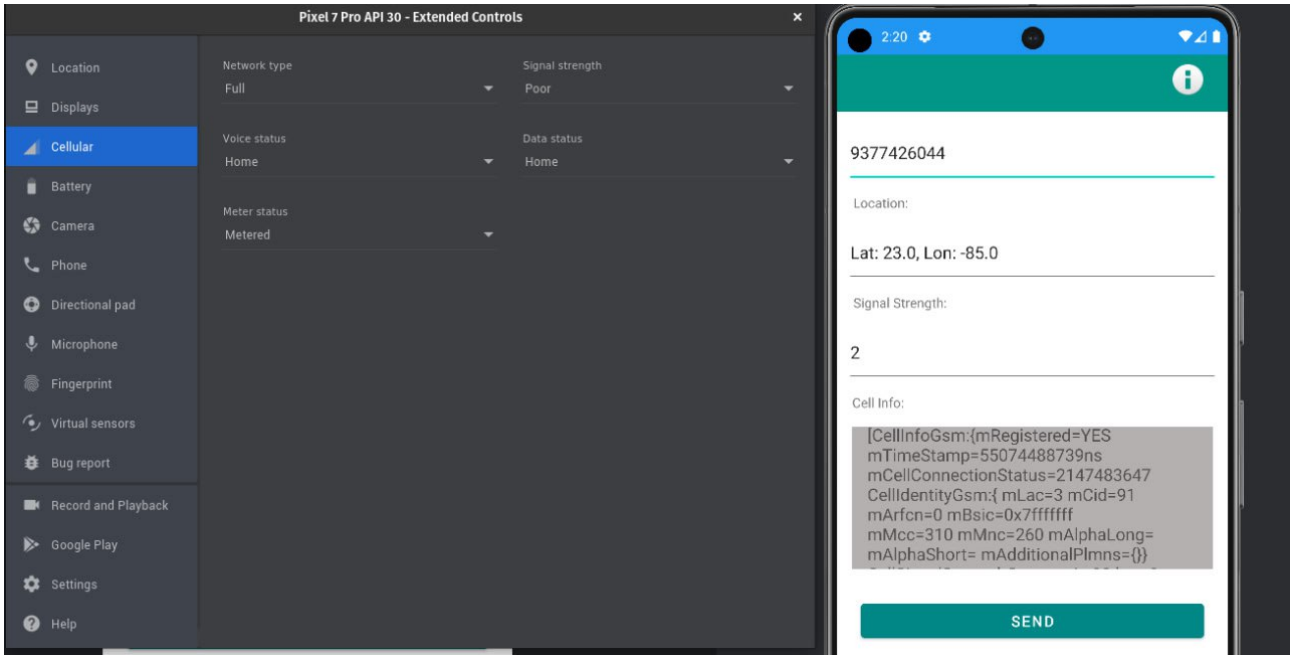
در این مرحله، بعد از فشردن دکمه ی start برنامه اندروید کلاینت قادر به شروع اندازه گیری سیگنال می شود و قدرت سیگنال را به صورت خودکار اندازه گیری کرده و به روزرسانی می‌کند. نمونه های تست شده در دستگاه های مختلف در ادامه آمده است.

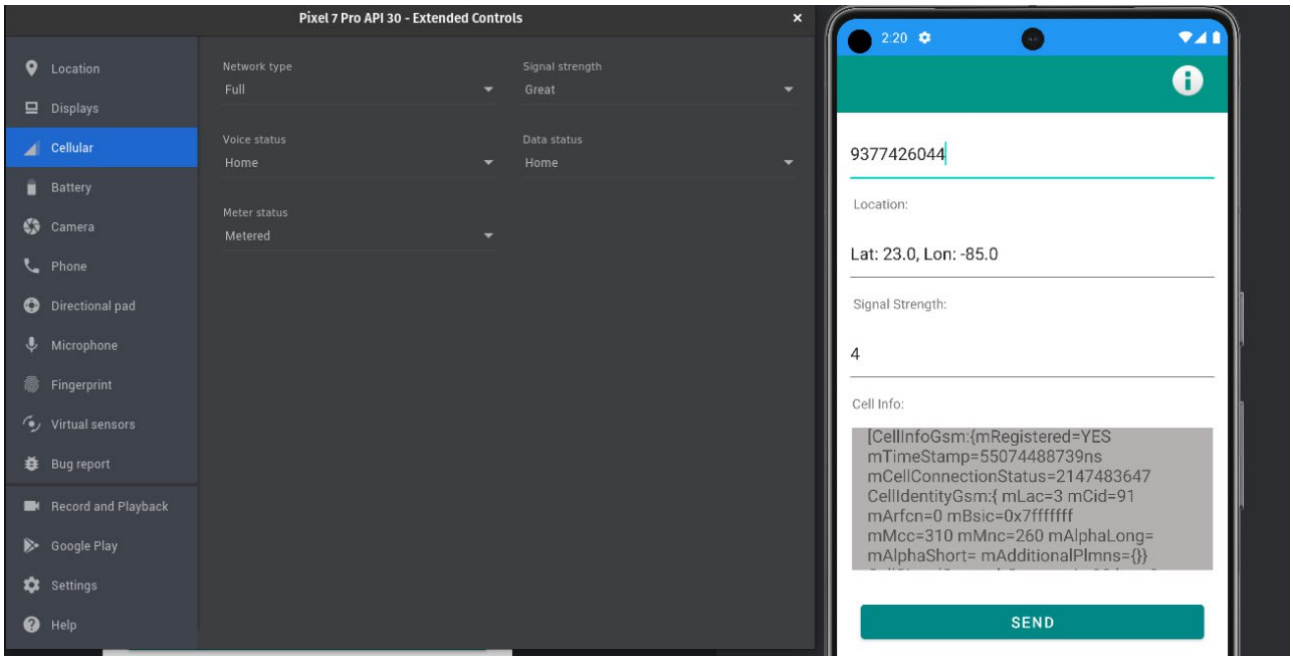


برای تست کردن تغیر وضعیت در سیمولیشن android studio می توانیم و بررسی صحت اطلاعات نرم افزار می توانیم اطلاعات سیستم را تغیر داده و نتیجه را به درستی در نرم افزار مشاهده کنیم .



در تصویر قبل صحت location و تصاویر بعد صحت تغیر شدت سیگنال از poor به great مشاهده می شود.

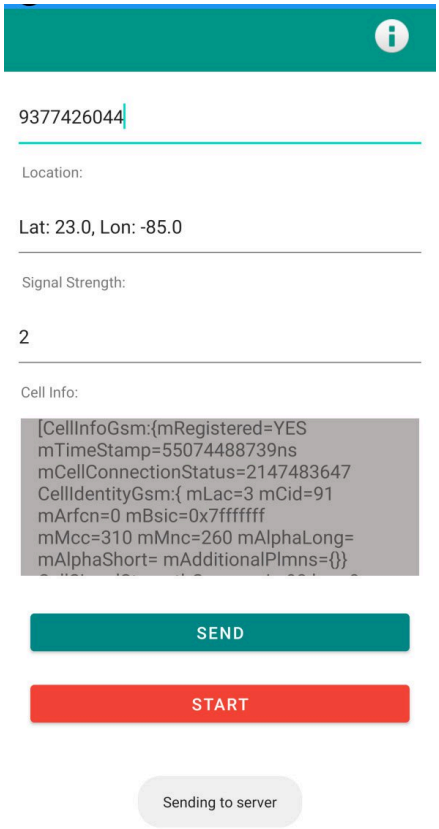




ارسال اطلاعات به سرور:

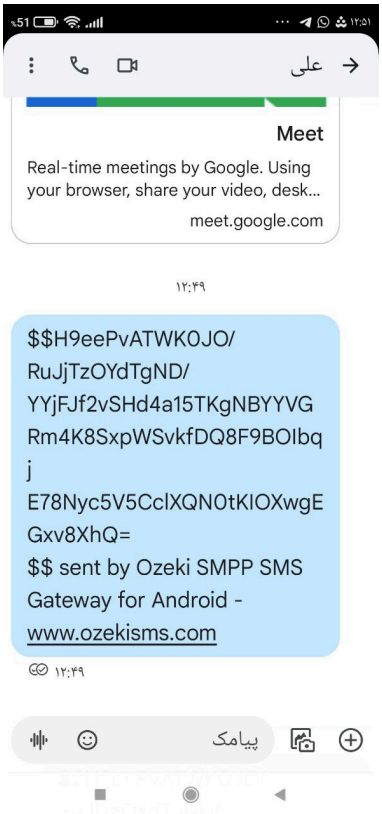
برنامه اندروید کلاینت هرگاه نیاز باشد، داده های اندازه گیری شده را از طریق پیامک (SMS) به سرور ارسال می‌کند، به ویژه زمانی که قدرت سیگنال به زیر یک حد مشخص کاهش یابد. همچنین با فشردن دکمه ی send هم کاربر می تواند این کار را انجام دهد.

در ادامه مشاهده می شود که هنگامی که سیگنال ضعیف شده است پیام sending to server نمایان می شود.



ارسال پیام کاربر به سرور:

کاربر یک پیام حاوی زمان ورود یا خروج خود را به صورت sms به سمت روتر ارسال می‌کند. این پیام پس از رمزگذاری توسط کلید داخلی برنامه (که یک کانال امن به حساب می‌آید)، به سرور فرستاده می‌شود.



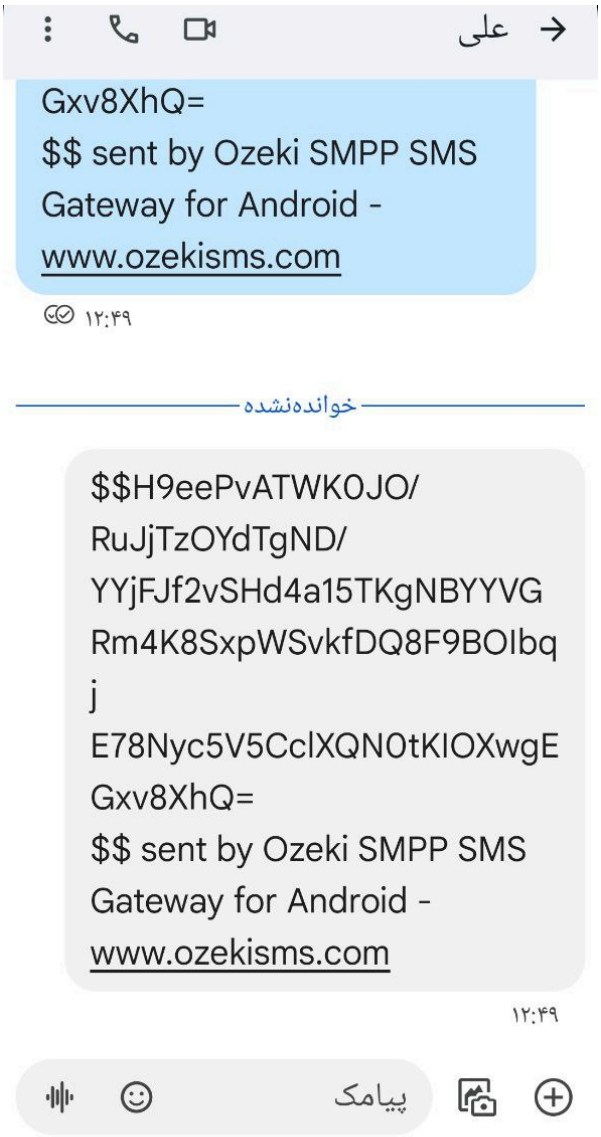
رمزگشایی و ذخیره پیام توسط سرور:

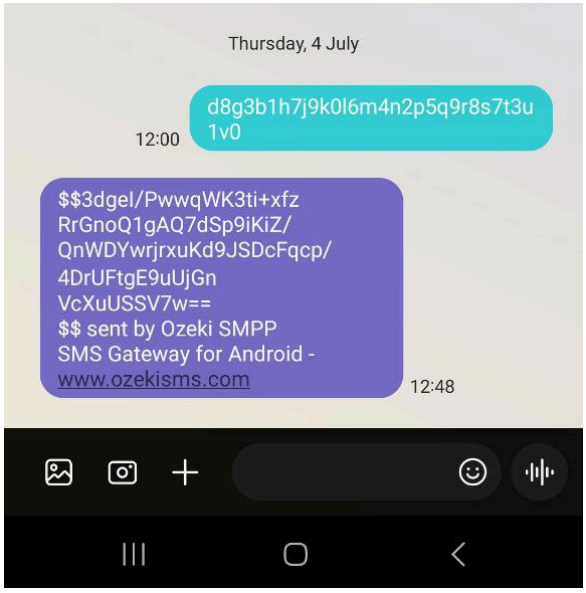
سرور پس از دریافت پیام، آن را رمزگشایی کرده و محتوای آن را ذخیره می‌کند.

تأیید دریافت پیام (ack):

سرور برای تأیید دریافت به برنامه گوشی (و کاربر)، ابتدا پیام را hash کرده و عبارتی با مقدار SHAKBAK به آن اضافه می‌کند و کل محتوا را رمزگذاری کرده و به گوشی ارسال می‌کند.

در ادامه پیام ارسال شده توسط سرور و دریافت شده در client را مشاهده می کنیم.





بررسی صحت پیام توسط گوشی:

هدف از hash کردن پیام این است که اگر در مسیر پیام تغییر کند یا توسط یک فرد میانی دستکاری شود، گوشی با بررسی تطابق hash پیام اولیه و hash دریافت شده، از این تغییر مطلع شده و پیام را مجدداً به سرور ارسال می‌کند.

تضمین امنیت پیام ها:

این روش تضمین می‌کند که محتوای اصلی پیام تغییری نکرده باشد. همچنین با رمزگذاری تمام پیام های رد و بدل شده بین دو طرف، امکان خواندن و یا تغییر پیام ها توسط فرد میانی را از بین می‌بریم.

این سناریو به برنامه اندروید کلاینت این امکان را می‌دهد که به طور مؤثر و کارآمد داده های سیگنال را مانیتور کرده و در مواقع ضروری به سرور ارسال کند تا از کیفیت ارتباطات اطمینان حاصل شود.