Service für Exposimeter

ServiveKlasse: CommunicationService.java

- Enthält Fake_TCP_Server, WifiDataBuffer, Broadcastsender.
- Darin müssen Remo & Matthü nichts machen.
- Funktioniert auch mit dem Nicht-Fake TCPServer :)

Jede ActivityKlasse muss folgendes enthalten, damit sie mit dem Service kommunizieren kann:

- **final String LOG_TAG = "NameOfThisActivity"; // alsAttribut,**
- **WifiDataBuffer wifiDataBuffer = new WifiDataBuffer(); // darin kann Broadcastreceiver die empfangenen Daten reinqueueun**
- **MyActivityReceiver myActivityReceiver = new MyActivityReceiver(LOG_TAG, wifiDataBuffer); // eben**
- MyActivityReceiver ist eine eigene Klasse
- Es git KEIN Attribut **Service service = new Service(); !!!**

```
onStart() {
        ...
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(CommunicationService.TRIGGER_Serv2Act);
        registerReceiver(myActivityReceiver, intentFilter);
        StartService(); // ist eine Funktion, siehe unten
        ...
        super.onStart();
}
```

```
private void StartService() {
    Intent intent = new Intent(this, CommunicationService.class);
    startService(intent);
}
```

senden von TriggerPacketen Activity → Service:

```
private void SendCaliTrigger() {
    byte[] calitrigger = {82, 68, 49, 54, 67, 65, 76, 68,0, 0, 0, 125, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 80, 69, 78, 68};
    sendTrigger(calitrigger); // see below
}
private void sendTrigger(byte[] TriggerPack) {

    Intent intent = new Intent();
    intent.setAction(CommunicationService.ACTION_FROM_ACTIVITY);
    intent.putExtra(CommunicationService.TRIGGER_Act2Serv, TriggerPack);
    sendBroadcast(intent);
}
```

Jede Activity, die einen Broadcastreceiver anlegt, muss diesen wieder abmelden:

```
public void onStop() {
      ...
      unregisterReceiver(myActivityReceiver);
      ....
      super.onStop();
}
```

Damit wird sichergestellt, dass es nur einen Receiver auf einmal gibt.

onDestroy() {..} UND onCreate() {..}: Nichts Servicelastiges!

// Bis hier ist, was in jeder Activity drinn stehen muss.

EINMAL im Ganzen App:

Service starten: Nicht nötig. Das StartService() im onStart() { } ruft nur einmal den Constructor und das onCreate() des Services auf. Jedes weitere StartService() ruft nur noch das onStartCommand() auf, welches nur überprüft, ob der Hotspot noch an ist.
Knacknuss: Service stoppen: Wäre schön, wenn folgender Code bei **Appschluss einmal** ausgeführt wird. Das AndroidOS killt zwar den Service (wenn das App geschlossen wird, zum Akkusparen), aber das onDestroy() des Services wird nicht aufgerufen. Im onDestroy wird der Hotspot aus- und das Wifi gegebenenfalls wieder eingeschaltet. Möglicherweise kann man im Eistellungsfenster einen Button “App schliessen” machen? Dieser Stopt den Service und schliesst das App mit allen runningen Activities??

```
private void StopService() {
      Log.d(LOG_TAG, "StopService called");
      Intent intent = new Intent();
      intent.setAction(CommunicationService.ACTION_FROM_ACTIVITY);
      intent.putExtra(CommunicationService.COMMAND_Act2Serv,
         CommunicationService.CMD_STOP);
      sendBroadcast(intent);
}
```

Noch eine Bemerkung zum Kallibrieren:

Der TCPServer empfangt die CALD-Packages in sechsundzwanzig 2048Bytes Chunks. Etwa alle Sekunde eines.

TCP_SERVER.getProgress() geht jetzt natürlich nicht mehr, da die Activities nicht mehr auf den TCPServer zugreifen können. ABER: Der TCPServer könnte bei empfangen eines Chunks ein Byte[] an den ActivityBroadcastReceiver senden.

RD16|PROG|int16|PEND mit Länge 14 Bytes
wobei der int16 den Fortschritt in % angiebt.

Dies sollte heute um 09:15 implementiert sein.