Interacting with the IRZ Radar 24 GHz via the IRZ Json Adaper

1 General information

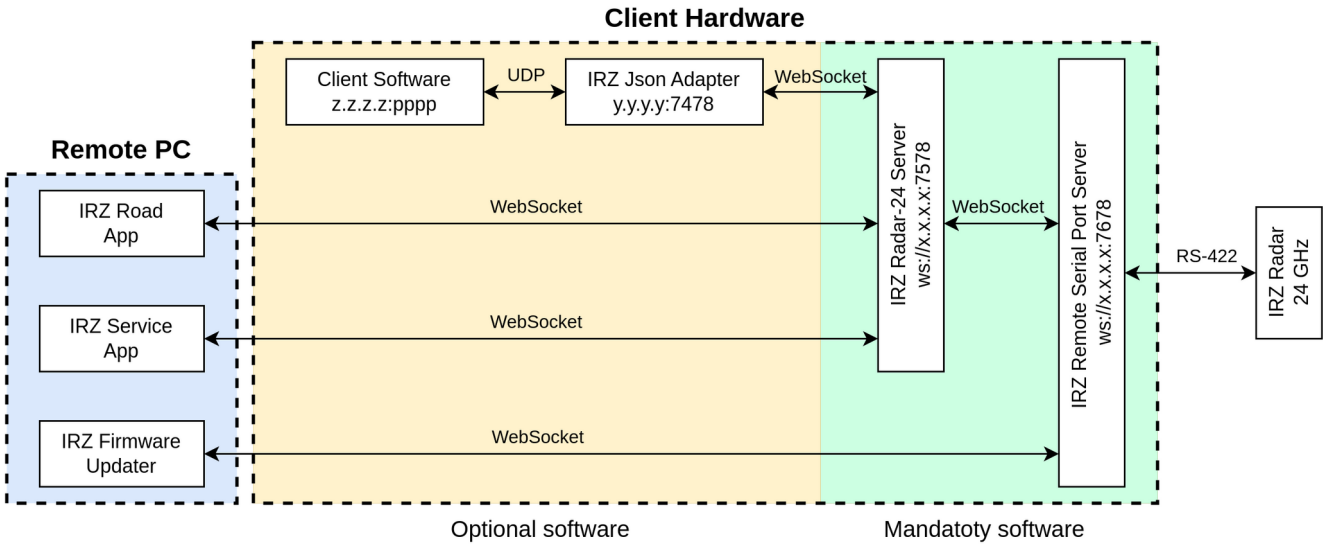

Figure 1 — Software interaction diagram

The IRZ Radar 24 GHz shall be connected to the client hardware via a digital UART-like **RS-422** line. Figure 1 shows software interaction diagram. The green filling indicates the software, installation of which is mandatory for functioning of the IRZ Json Adapter, as well as for providing the possibility of remote reprogramming of the IRZ Radar 24 GHz. Yellow colour denotes software that implements software interacting process described herein.

2 Connecting to the IRZ Json Adapter

Since Udp transport does not involve establishing a connection, the IRZ Json Adapter stores in its configuration a list of all clients to which packets from the radar are sent. The client software should also store the adapter's address in its configuration. Typically (see Figure 1), the adapter is available at **y.y.y.y.y:7478** and the client software can use any Udp address **z.z.z.z.z.z:pppp**.

The IRZ Json Adapter converts binary data received from the radar into a Json text representation, and translates text commands from the client software back into binary commands, which are transmitted to the device via the IRZ Radar-24 Server.

3 Json message format

There are three basic types of Json messages to interact with the IRZ Radar 24 GHz:

1) Radar state data (from the radar to the client software only);
2) Regular data on road objects (from the radar to the client software only);
3) Radar setup commands:
   a) Requests (from the client software to the radar);
   b) Response (from the radar to the slient software).

Note: Parameters in Json messages can be arranged in any order, as this is not specified in the standard. Some software implementations order them alphabetically.

3.1 Radar state data (Json message, type 1)

This message is sent each time the radar status changes and periodically at 5-second intervals. Message structure with example data is as follows:

```
{
  "name": "STATE",
  "state_code": 2,
  "state_time": "2024-09-26T09:20:05.625+04:00",
  "sensor_id": "radar id"
}
```

Table 1 — Parameters of radar state data

Parameter	Data type	Possible values	Units	Notes
state_code	integer enum	{0, 1, 2}, {–1}		radar state code: **0** — no messages (radar is not connected, not powered on, is starting or restarting now); **1** — radar is busy (initialising or changing its operating mode); **2** — radar is working and ready to receive commands; **–1** — IRZ Json Adapter is not configured properly (radar uses a different binary protocol)
state_time	string			ISO 8601 compliant date&time of state change
sensor_id	string			radar id string

3.2 Regular data on road objects (Json message, type 2)

Message structure with example data is as follows:


```
{
  "name": "OBJECTS",
  "protocol_version": "1.0",
  "cycle_id": 11965,
  "frame_time": "2024-09-26T09:23:31.795+04:00",
  "rows": 2,
  "rows_data": [
    {
      "sensor_id": "SensR-24.01 2201 000005",
      "time": "2024-09-26T09:23:31.795+04:00",
      "obj_id": 35,
      "lane": 2,
      "obj_class": "B",
      "obj_length": 4.4,
      "point_x": 22.56,
      "point_y": -3,
      "obj_speed": 6.48,
      "obj_speed_mps": 1.8,
      "heading": 0
    },
    {
      "sensor_id": "SensR-24.01 2201 000005",
      "time": "2024-09-26T09:23:31.645+04:00",
      "obj_id": 42,
      "lane": 5,
      "obj_class": "E",
      "obj_length": 18,
      "point_x": 43.72,
      "point_y": 11.4,
      "obj_speed": -36,
      "obj_speed_mps": -10,
      "heading": 180
    }
  ]
}
```

Table 2 — Parameters of regular data on road objects

Parameter	Data type	Possible values	Units	Notes
protocol_version	string	"1.0"		Json protocol version
cycle_id	integer	[0; 2^{32}–1]		cycle sequential number[1]
frame_time	string			ISO 8601 compliant date&time when frame received
rows	integer	[0; 64]		number of records in the array of road objects
rows_data	object array			array of road objects
sensor_id	string			radar id string
time	string			ISO 8601 compliant date&time of last detection[2] of road object
obj_id	integer	[0; 63]		road object id
lane	integer	[0; 7], {–1}		lane number starting from **0**; **–1** — lane number is undefined[3]
obj_class	string	{"A", "B", "C", "E"}, {"N"}		Vehicle class: **"A"** — motorbike, quad bike, moped, bicycle; **"B"** — private car; **"C"** — truck / lorry; **"E"** — long freight vehicle, trailer or road train; **"N"** — vehicle class is undefined
obj_length	float		m	road object length
point_x	float		m	**X**-axis coordinate of the road object (along the road)[4]
point_y	float		m	**Y**-axis coordinate of the road object (across the road)[4]
obj_speed	float	(–360; +360)	km/h	road object speed[4]
obj_speed_mps	float	(–100; +100)	m/s	road object speed[4]
heading	float	(-180; 180]	ang. degrees	road object azimuth[5]

[1] The **cycle_id** parameter is an incremental counter. When it overflows, its value is reset to zero. When radar is busy (see STATE message), the counter value is not incremented.

[2] If road object is detected in the current cycle, the value of its **time** parameter is equal to the **frame_time** parameter value of the whole frame, otherwise the road object is sent by the IRZ Json Adapter for another **500 ms** to avoid 'flickering' (disappearance and reappearance), prediction of movement is not performed.

[3] Splitting road into lanes is performed by accumulating and processing the statistics of road object passages during the first **24 hours** from the moment the radar is powered on or restarted. Lane number detection becomes available not earlier than after **1 hour** of operation from the moment the radar is powered on or restarted. **Note:** this feature is under development yet.

[4] If radar is mounted over the axis (median strip) of a right-hand traffic road, then outgoing objects have a negative **Y** coordinate and a positive speed, and vice versa for oncoming objects. The **X**

coordinate of the road objects is always positive (it could be negative only if the **X** coordinate of the radar itself is set to a negative value, see SET_POSITION message).

[5] The value of the **heading** parameter is approximate, **0 degrees** is for the outgoing traffic, while **+180 degrees** is for oncoming.

3.3 Radar setup commands (Json message, type 3)

Note: It is recommended to send setup commands sequentially, proceeding to the next one only after receiving a response about successful completing of the current one. Due to the fact that some text commands (for example, SET_LIMITATIONS) are converted into more than ten binary commands, the waiting time for a response message to be received can be up to one minute. The **sensor_id** parameter is not used yet and can be left empty or omitted in all command requests.

All setup commands (type 3, a) are answered with the same type of response message (type 3, b), and the **name** parameter of the response matches the same parameter of the sent command request. Response message structure with example data is as follows:

```
{
  "name": "SET_POSITION",
  "count": 1,
  "data": [
    {
      "result": true,
      "sensor_id": "SensR-24.01 2201 000005"
    }
  ]
}
```

Table 3 — Parameters of a command response

Parameter	Data type	Possible values	Units	Notes
count	integer	1		always 1
data	object array			array of received data with the only element
result	bool			command execution result: **true** — command succeeded; **false** — command failed
sensor_id	string			radar id string

3.3.1 Command to set radar position

Message (type 3, a) structure with example data is as follows:

```
{
  "name": "SET_POSITION",
  "count": 1,
  "data": [
    {
      "x": 0,
      "y": 4,
      "z": 5.2,
      "xy": -7.5,
      "xz": 9.1,
      "yz": 0,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 4 — Command parameters to set radar position

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
x	float	[–25; 25]	m	**X**-axis offset[1] (usually 0)
y	float	[–25; 25]	m	**Y**-axis offset[1]
z	float	[0; 20]	m	mounting height[1]
xy	float	[–30; 30]	ang. degrees	azimuth (jaw) angle[1]
xz	float	[–30; 30]	ang. degrees	elevation (pitch) angle[1]
yz	float	[–30; 30]	ang. degrees	bank (roll) angle[1] (usually 0)
sensor_id	string			radar id string (not used)

[1] The **X**, **Y** and **Z** axes of the road form the right-hand Euclidean coordiate system. If radar is mounted over the axis (median strip) of a right-hand traffic road, then the azimuth (jaw) angle is positive if the radar normal is rotated towards the oncoming traffic lanes (counterclockwise when viewed from the end of the **Z** axis). The elevation (pitch) angle is positive if the radar normal is inclined downwards from the horizontal position (counterclockwise when viewed from the end of the **Y** axis), see Figure 2. The bank (roll) angle is positive if the radar is rotated counterclockwise when viewed from the end of the **X** axis.

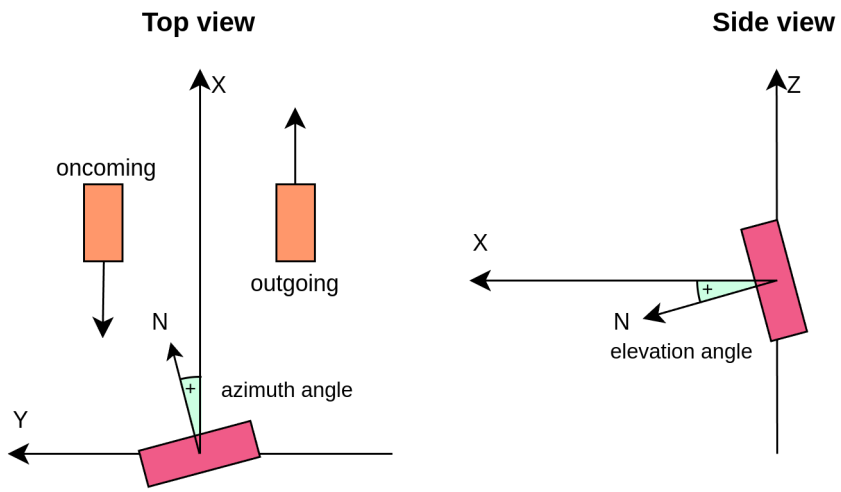

Figure 2 — Radar axes

Note: Radar position is not saved to flash automatically after setting by a command request. To save the position and other parameters into the flash memory, please use RESTART_RADAR command with the **save_to_flash** parameter set to **true**.

3.3.2 Command to switch radar operating mode

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "SET_MODE",
  "count": 1,
  "data": [
    {
      "mode": 0,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 5 — Command parameters to switch radar operating mode

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
mode	integer enum	{0, 1, 2}		operation mode to switch radar to[1]: **0** — normal (road) mode; **1** — verification mode, simulator is "IS-24" (24087 MHz); **2** — verification mode, simulator is "Sapsan 3M lit. 2" (24000 MHz)
sensor_id	string			radar id string (not used)

[1] Radar always starts operating in the normal (road) mode. Switching to the verification mode automatically terminates the fake targets (demo road objects) mode (see SET_FAKE_TARGETS message) and disables the limitations on road objects issuing (see SET_LIMITATIONS message). After switching back to the normal (road) mode, fake targets mode and limitations on road objects issuing should be manually reactivated, if needed.

3.3.3 Command to switch fake targets mode

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "SET_FAKE_TARGETS",
  "count": 1,
  "data": [
    {
      "enabled": true,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 6 — Command parameters to switch fake targets mode

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
enabled	bool			fake targets (demo road objects) mode: **true** — switch on[1]; **false** — switch off
sensor_id	string			radar id string (not used)

[1] The switched on state of the fake targets (demo road objects) mode is not saved to flash and will not be restored after restarting the radar.

3.3.4 Command to configure limitations on road objects issuing

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "SET_LIMITATIONS",
  "count": 1,
  "data": [
    {
      "spd_x_min": 1,
      "spd_x_max": 100,
      "spd_y_min": 0,
      "spd_y_max": 10,
      "pos_x_min": 5,
      "pos_x_max": 100,
      "pos_y_min": -10,
      "pos_y_max": 10,
      "enabled": true,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 7 — Command parameters to configure limitations on road objects issuing

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
spd_x_min	float	[0; 10]	m/s	**X**-axis minimum speed (modulo)
spd_x_max	float	[10; 100]	m/s	**X**-axis maximum speed (modulo)
spd_y_min	float	[0; 10]	m/s	**Y**-axis minimum speed (modulo)
spd_x_max	float	[10; 100]	m/s	**Y**-axis maximum speed (modulo)
pos_x_min	float	[0; 50]	m	**X**-axis minimum position (always ≥ 0)
pos_x_max	float	[50; 250]	m	**X**-axis maximum position (always > 0)
pos_y_min	float	[–25; 25]	m	**Y**-axis minimum position (signed)
pos_y_max	float	[–25; 25]	m	**Y**-axis maximum position (signed)
enabled	bool			limitations on road objects issuing: **true** — switch on; **false** — switch off
sensor_id	string			radar id string (not used)

3.3.5 Command to set radar sensitivity threshold

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "SET_SENSITIVITY",
  "count": 1,
  "data": [
    {
      "threshold": 100,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 8 — Command parameters to set radar sensitivity threshold

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
threshold	integer	[1; 500]		radar sensitivity threshold[1]
sensor_id	string			radar id string (not used)

[1] Metal objects, fences, billboards, hanging road signs, etc. located in front of the radar may cause false detections. The higher the value of the **threshold** parameter, the more noise and interference are suppressed, which leads to a reduction in the range of detection of road objects and missing some of them in the radar output. The closer to the road surface the radar is mounted, the higher the sensitivity threshold value should be set. It's preferable to adjust this parameter at each specific location where the radar is mounted. The general recommendation for picking the sensitivity threshold is as follows: increase the value until the detection range drops below the required value or until road objects without a track begin to appear frequently in the far lanes, and then decrease the threshold back by **20–40** points.

3.3.6 Command to select frequency channel

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "SET_CHANNEL",
  "count": 1,
  "data": [
    {
      "channel_id": 1,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 9 — Command parameters to select frequency channel

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
channel_id	integer	[0; 16]		frequency channel number[1]
sensor_id	string			radar id string (not used)

[1] If there is another 24 GHz radar operating nearby, distorted tracks and false detections may appear, especially if the radars are directed towards each other. If the radar regularly gives false detections at certain points or zones, i.e. objects that do not move from their place but keep the same speed, there is probably a frequency conflict with another radar. In this case, it's recommended to empirically pick a frequency channel on which the number of false detections is minimal. If both radars are manufactured by IRZ, then channels should be chosen from the **{0, 8, 16}** series, since these channels are completely separated in pairs by the emitted signal frequency.

3.3.7 Command to send free binary set command

Note: This command should only be used when there is no dedicated Json command to request some required operation.

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "FREE_SET_COMMAND",
  "count": 1,
  "data": [
    {
      "is_fixed": false,
      "action": 148,
      "param_number": 4,
      "param_value": 150,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 10 — Command parameters to send free binary set command

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
is_fixed	bool			data type[1] of the parameter to be set: **true** — fixed type; **false** — integer type
action	integer	[0; 255]		action number[1]
param_number	integer	[0; 255]		parameter number[1]
param_value	integer / float			parameter value: if type is **integer** — transmitted to the radar as is; if type is **fixed** — received float value is converted to a fixed-precision number with six decimal places (representation precision is $\mathbf{10^{-6}}$)
sensor_id	string			radar id string (not used)

[1] For data type, action number and parameter number, see Part 2 of the IRZ Radar 24 GHz Operation Manual.

3.3.8 Command to restart radar and save parameters to flash

Command requst message (type 3, a) structure with example data is as follows:

```
{
  "name": "RESTART_RADAR",
  "count": 1,
  "data": [
    {
      "save_to_flash": false,
      "sensor_id": "radar id"
    }
  ]
}
```

Table 11 — Command parameters to restart radar and save parameters to flash

Parameter	Data type	Acceptable values	Units	Notes
count	integer	1		always 1
data	object array			array of transmitted data with the only element
save_to_flash	bool			saving current parameters to the flash memory: **true** — radar will save its current parameters into the flash memory and restart; **false** — radar will restart, resetting its parameters to the previously saved values
sensor_id	string			radar id string (not used)