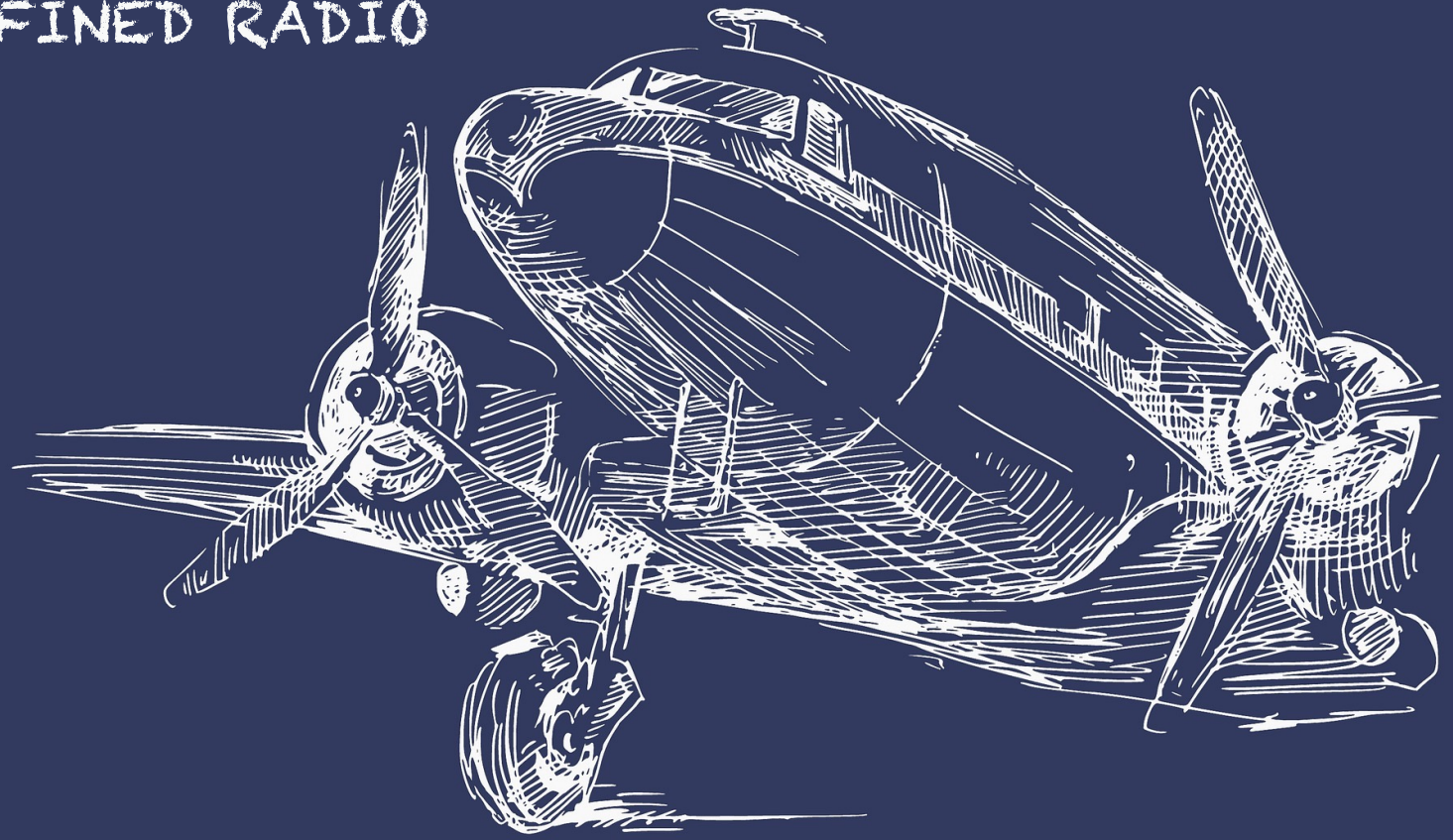
TRACKING AIRCRAFT

WITH REDIS + SOFTWARE-DEFINED RADIO



redis

Guy Royse

Developer Advocate

redis

@guyroyse

github.com/guyroyse

guy.dev


I AM NOT AN EXPERT


HERE THERE BE DEMOS

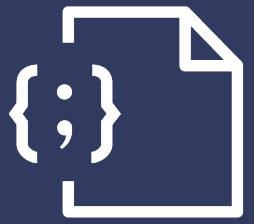

THIS TALK IS GONNA BE
ALL OVER THE PLACE

AIRCRAFT

RADIO

MICROSERVICES

STREAMS



REDISJSON

REDISEARCH


WHAT IS
SOFTWARE-DEFINED
RADIO?
redis
transistor

A REGULAR RADIO
BUT WITH LESS STUFF

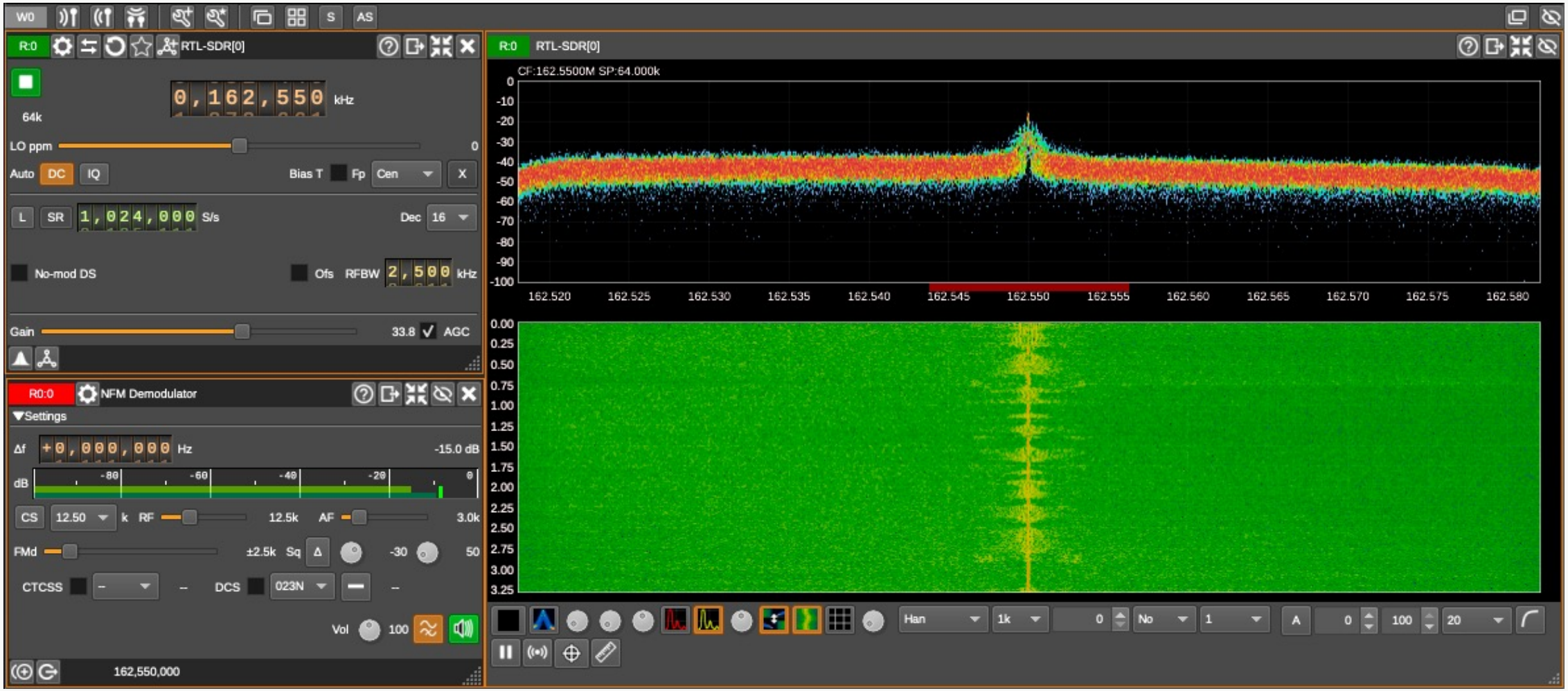
A RADIO YOU PLUG INTO YOUR COMPUTER

USB

PROPER LABELING

ANTENNA

SOFTWARE DOES THE REST



DEMO

redis

GETTTING THE HARDWARE

RTL-SDR

Upconverter

HF Antenna

SOFTWARE OPTIONS

Gqrx SDR

Open source software defined radio by Alexandru Csete OZ9AEC

CubicSDR

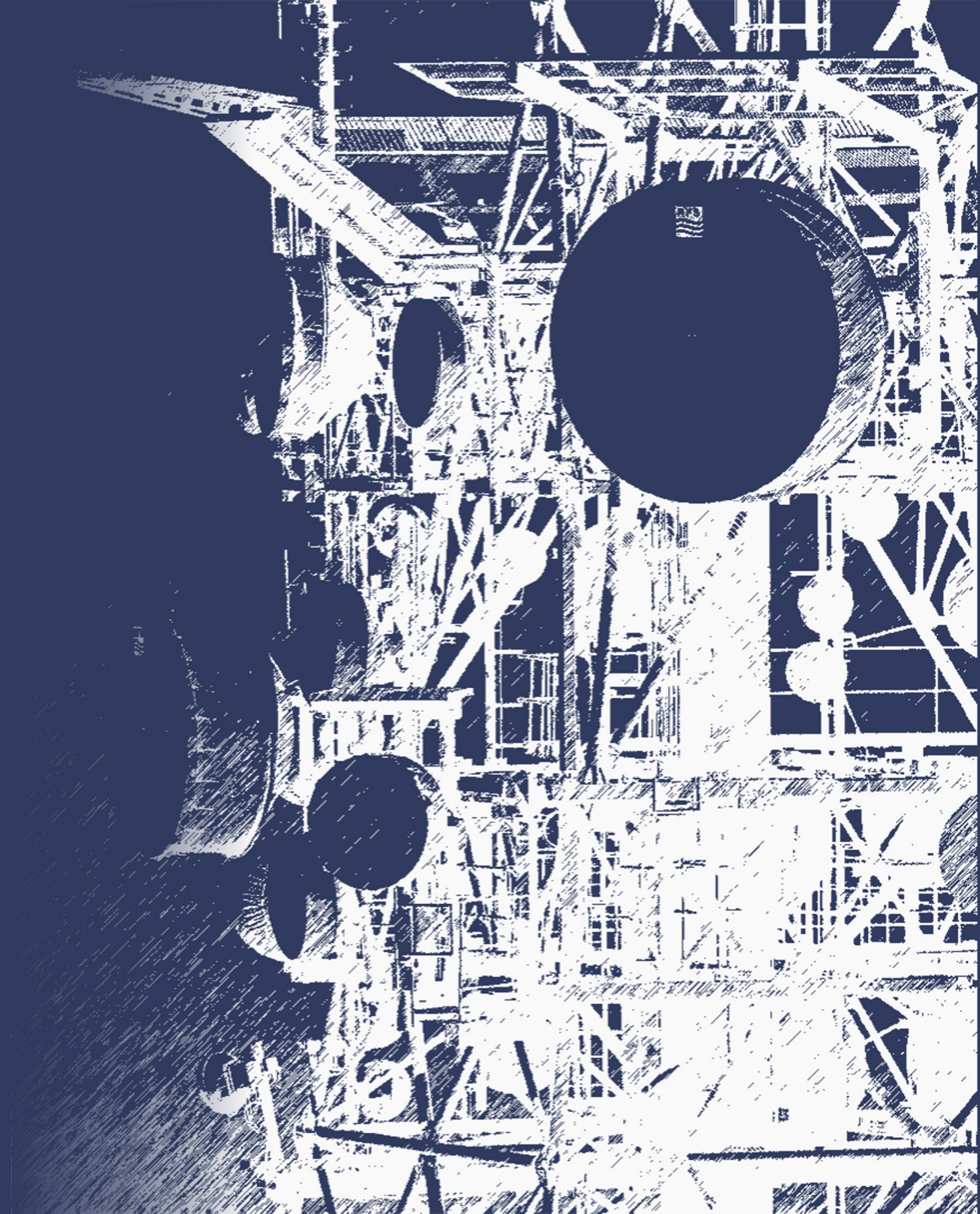
Cross-Platform and Open-Source Software-Defined Radio Application

DATA FORMATS

MORSE CODE

RTTY
RADIO TELETYPE

APRS
AUTOMATIC PACKET REPORTING SYSTEM



AUTOMATIC PACKET REPORTING SYSTEM

VEHICLES

REPEATERS

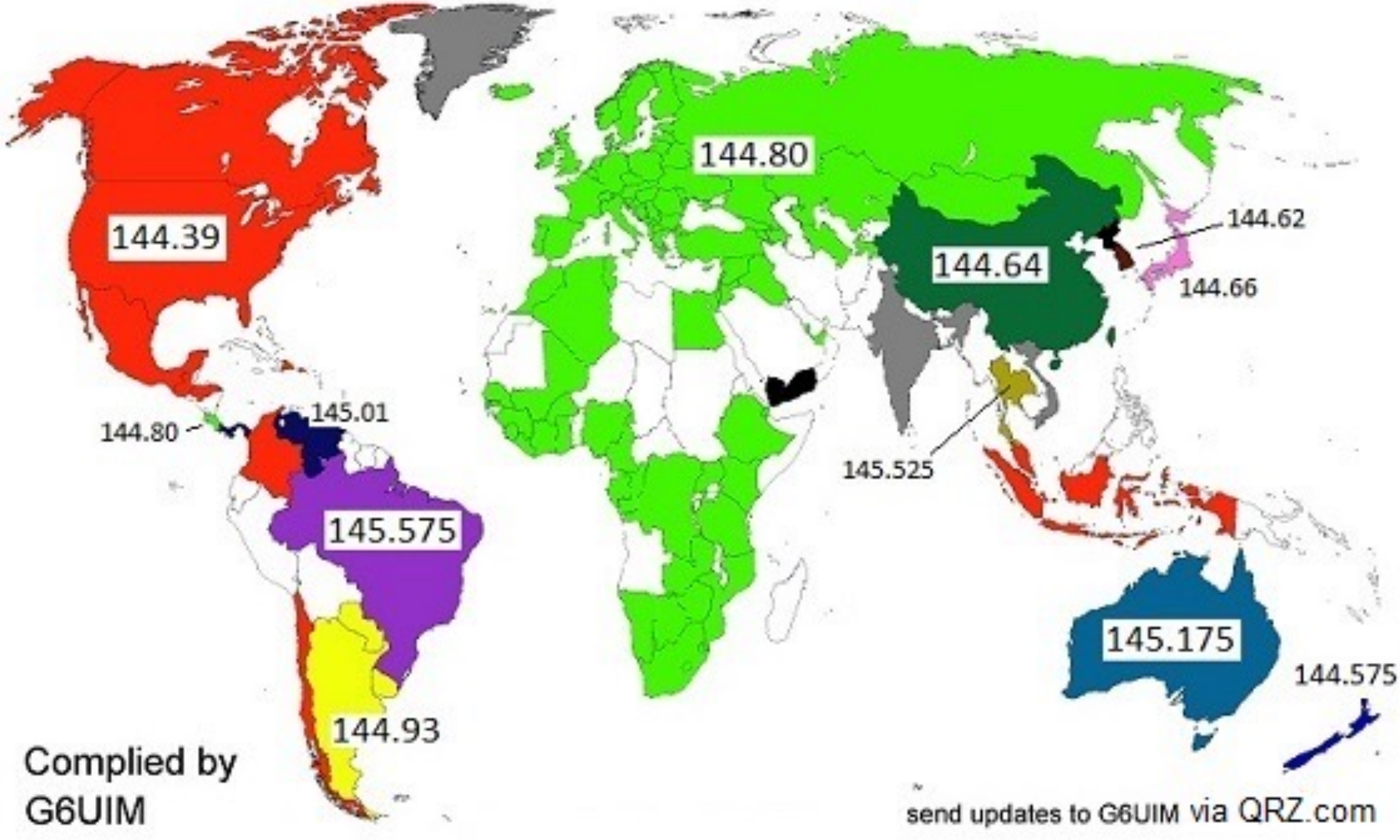
WEATHER STATIONS

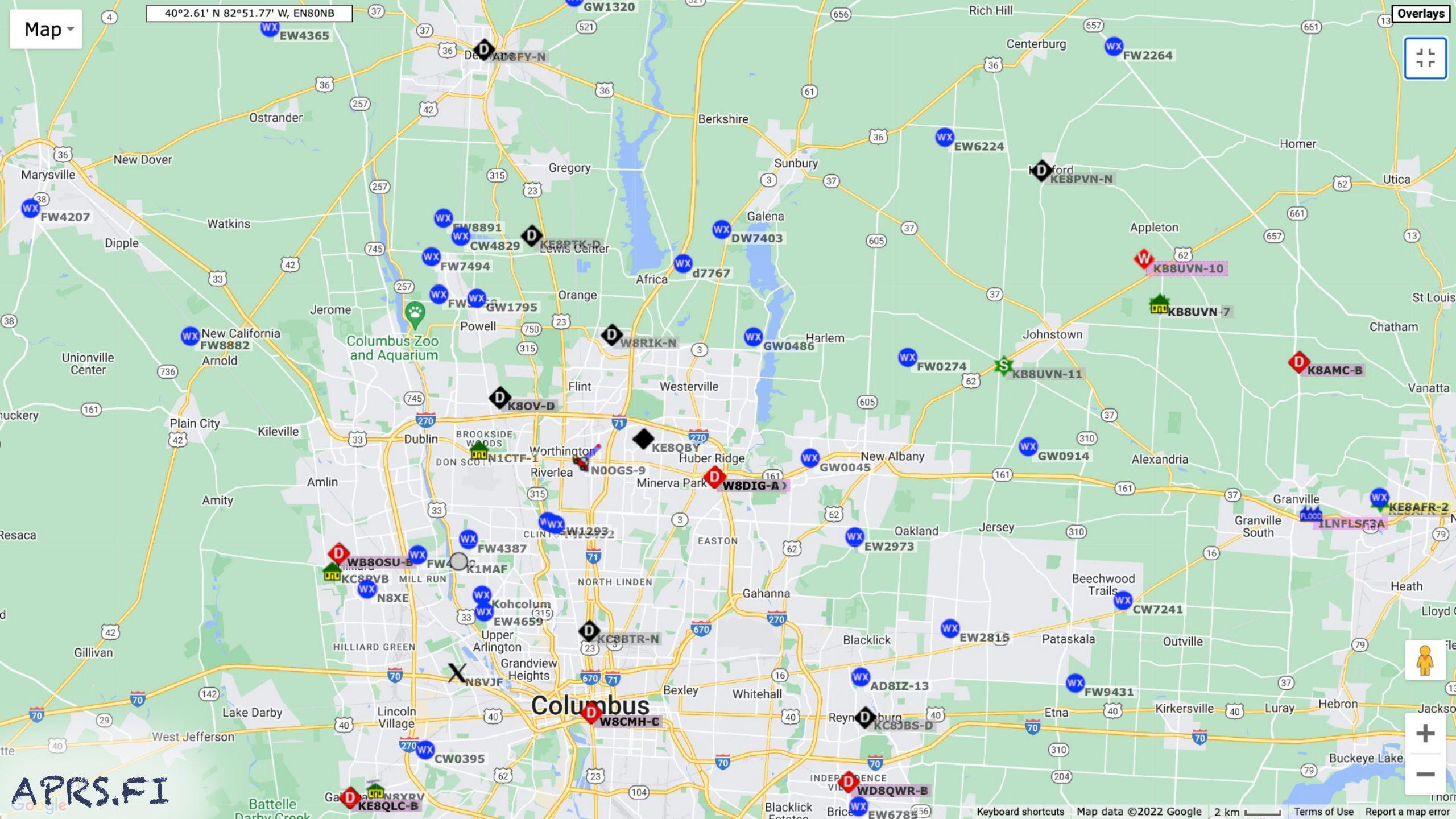
INDIVIDUALS

INDIVIDUALS

REPEATERS

INTERNET GATEWAYS


144.80
144.39
144.64
144.62
144.66
144.80
145.01
145.525
145.575
145.175
144.575
144.93
Complied by
G6UIM
send updates to G6UIM via QRZ.com


Map
40°2.61' N 82°51.77' W, EN80NB
Overlays
EW4365
GW1320
AD8FY-N
FW2264
Centerburg
Rich Hill
Ostrander
Berkshire
EW6224
Homer
Utica
New Dover
Marysville
Gregory
Sunbury
KE8PVN-N
FW4207
Watkins
Dipple
FW8891
CW4829
KE8PTK-D
Lewis Center
Galena
DW7403
Appleton
FW7494
d7767
Africa
KB8UVN-10
Orange
GW1795
KB8UVN-7
St Louis
Jerome
Powell
W8RIK-N
Chatham
New California
FW8882
Columbus Zoo
and Aquarium
GW0486
Harlem
Johnston
Arnold
Unionville
Center
FW0274
KB8UVN-11
K8AMC-B
Flint
Westerville
Vanatta
K8OV-D
Plain City
Kileville
Dublin
BROOKSIDE
WOODS
KE8QBY
N1CTF-1
Worthington
Huber Ridge
GW0914
Alexandria
Riverlea
N0OGS-9
New Albany
GW0045
Minerva Park
W8DIG-A
Amlin
Amity
Granville
KE8AFR-2
Granville
South
ILNFLS53A
Oakland
Jersey
EW2973
W1293
FW4387
EASTON
W8OSU-B
K1MAF
KC8RVB
MILL RUN
NORTH LINDEN
Beechwood
Trails
CW7241
N8XE
Kohcolum
EW4659
Gahanna
Heath
KC8BTR-N
EW2815
Blacklick
Pataskala
Outville
Gillivan
HILLIARD GREEN
N8VJF
Grandview
Heights
Upper
Arlington
AD8IZ-13
FW9431
Bexley
Whitehall
Columbus
W8CMH-C
Lake Darby
Lincoln
Village
Etna
Kirkersville
Luray
Hebron
KC8JBS-D
West Jefferson
CW0395
Buckeye Lake
INDEPENDENCE
WD8QWR-B
KE8QLC-B
N8XRV
Battelle
Darby Creek
EW6785
APRS.FI
Keyboard shortcuts
Map data ©2022 Google
2 km
Terms of Use
Report a map error

DEMO

redis

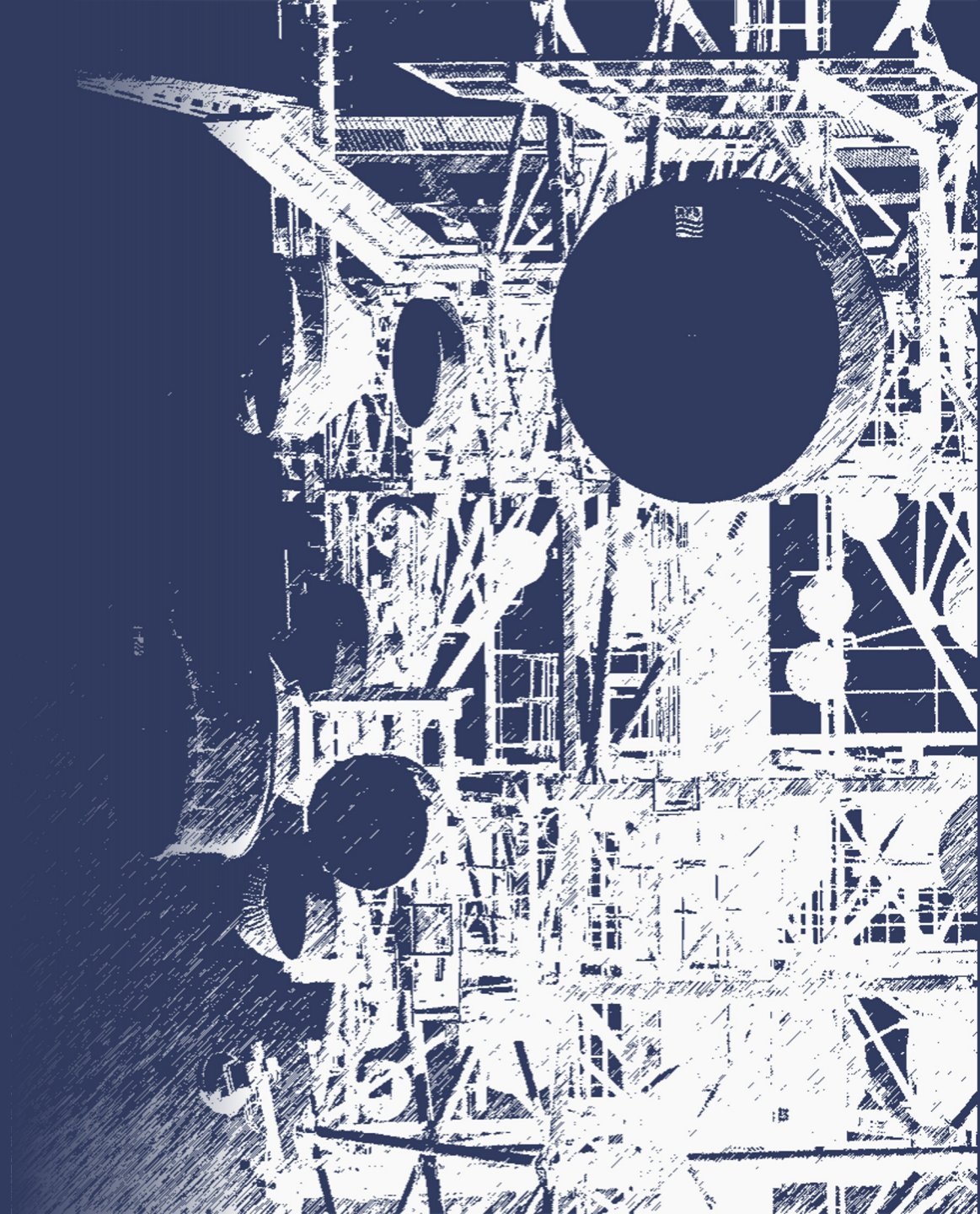
DATA FORMATS

MORSE CODE

RTTY
RADIO TELETYPE

APRS
AUTOMATIC PACKET REPORTING SYSTEM

AIS
AUTOMATIC IDENTIFICATION SYSTEM



AIS DATA

POSITION

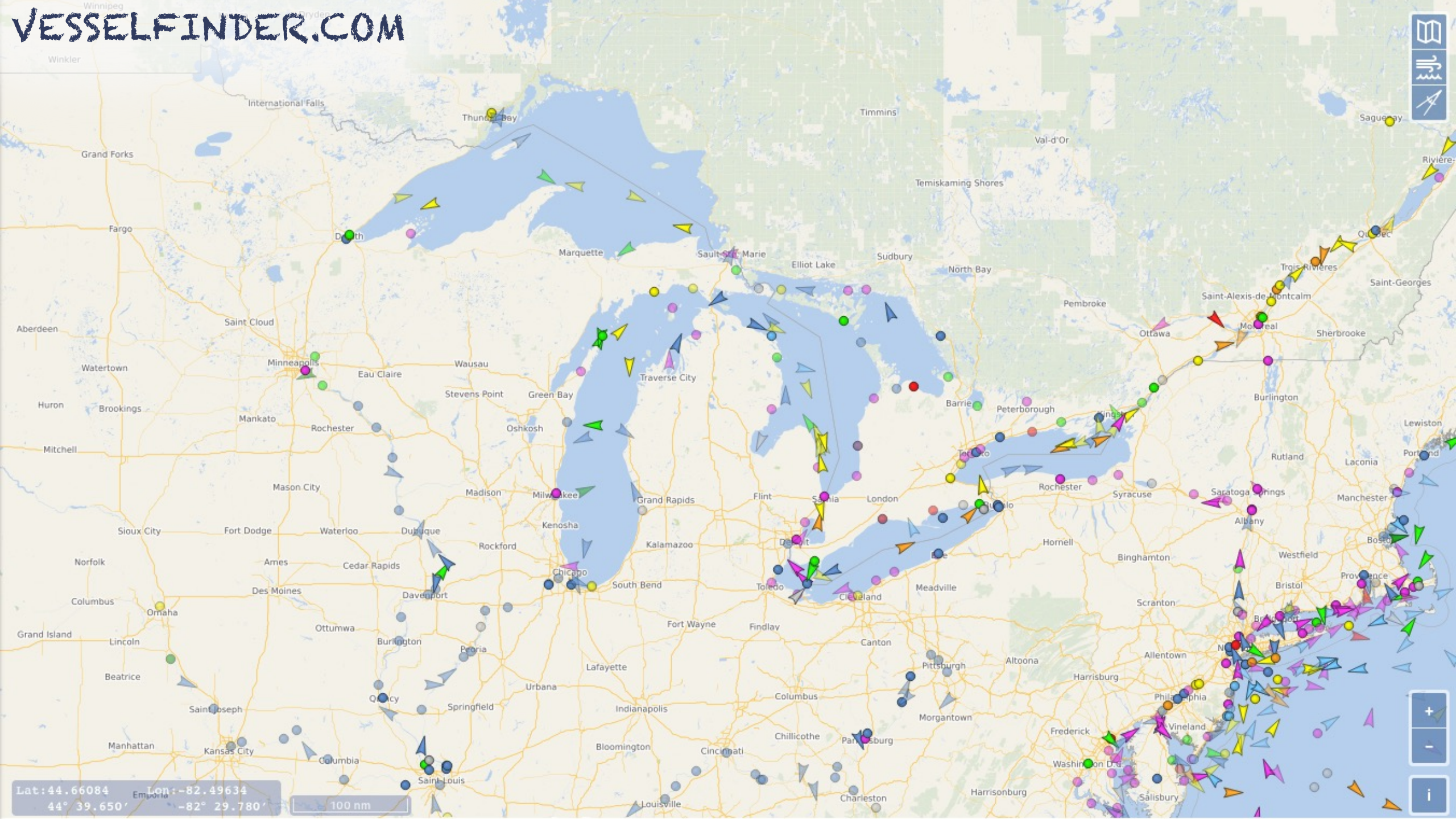
LATITUDE, LONGITUDE,
HEADING, SPEED

FREQUENCIES

161.975 MHz
162.025 MHz

SITES

VESSELFINDER.COM
MARINETRAFFIC.COM


VESSELFINDER.COM
Winnipeg
Winkler
International Falls
Thunder Bay
Timmins
Saguenay
Grand Forks
Val-d'Or
Temiskaming Shores
Fargo
Duluth
Marquette
Sault Ste. Marie
Elliot Lake
Sudbury
North Bay
Trois-Rivières
Saint-Georges
Pembroke
Saint-Alexis-de-Montcalm
Ottawa
Montreal
Sherbrooke
Aberdeen
Saint Cloud
Watertown
Minneapolis
Eau Claire
Wausau
Traverse City
Huron
Brookings
Stevens Point
Green Bay
Barrie
Peterborough
Burlington
Mankato
Rochester
Oshkosh
Mitchell
Rutland
Laconia
Lewiston
Mason City
Madison
Milwaukee
Grand Rapids
Flint
London
Rochester
Syracuse
Saratoga Springs
Manchester
Albany
Sioux City
Fort Dodge
Waterloo
Dubuque
Kenosha
Kalamazoo
Hornell
Binghamton
Westfield
Norfolk
Ames
Cedar Rapids
Rockford
Chicago
Bristol
Columbus
Des Moines
Davenport
South Bend
Toledo
Meadville
Scranton
Omaha
Ottumwa
Fort Wayne
Findlay
Grand Island
Lincoln
Burlington
Canton
Peoria
Pittsburgh
Altoona
Allentown
Beatrice
Lafayette
Harrisburg
Urbana
Columbus
Quincy
Springfield
Indianapolis
Morgantown
Saint Joseph
Frederick
Manhattan
Kansas City
Bloomington
Cincinnati
Chillicothe
Columbia
Saint Louis
Harrisonburg
Salisbury
Louisville
Charleston
Lat:44.66084 Lon:-82.49634
44° 39.650' -82° 29.780'
100 nm

DEMO

redis

DATA FORMATS

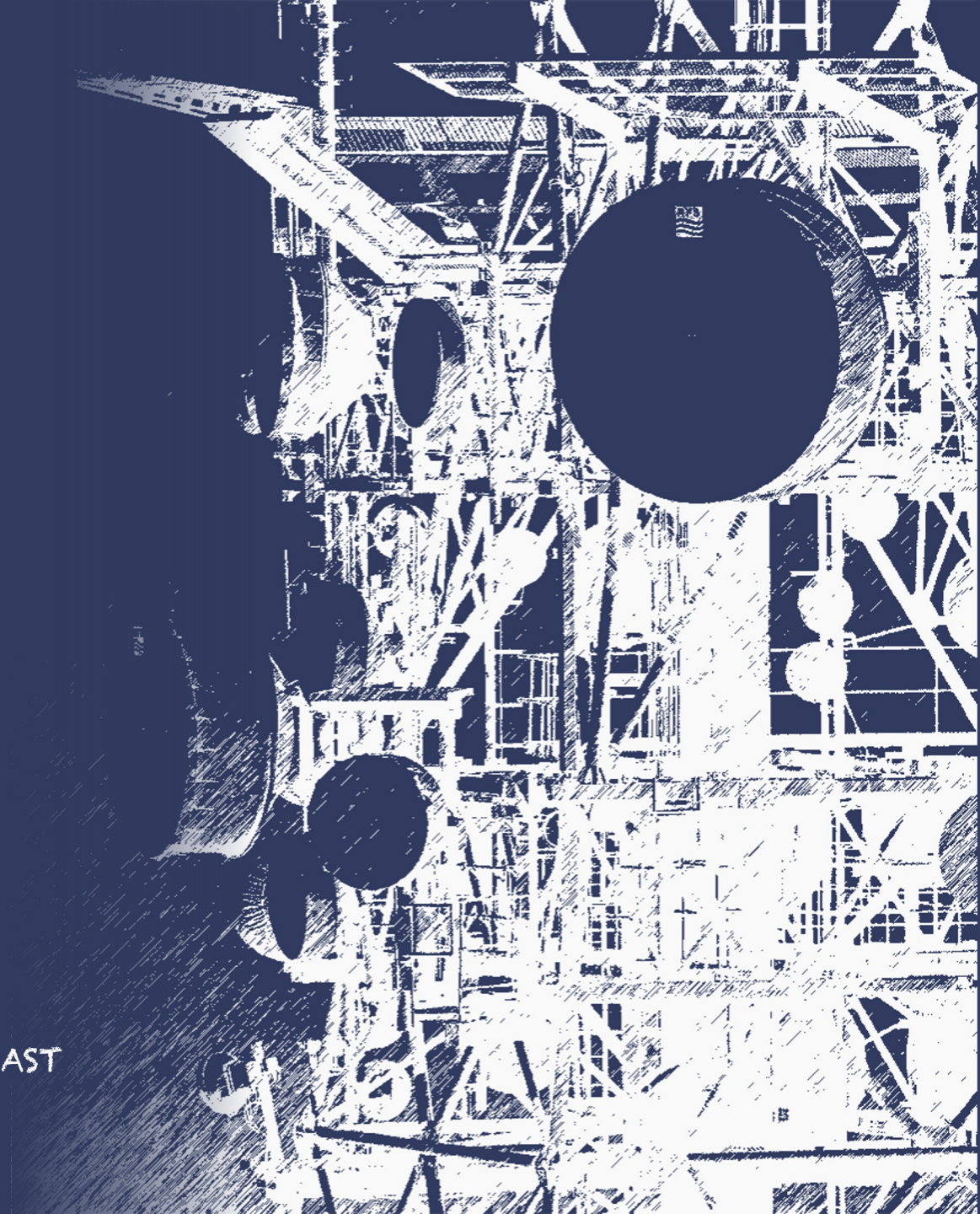
MORSE CODE

RTTY
RADIO TELETYPE

APRS
AUTOMATIC PACKET REPORTING SYSTEM

AIS
AUTOMATIC IDENTIFICATION SYSTEM

ADS-B
AUTOMATIC DEPENDENT SURVEILLANCE BROADCAST



1090 MHz

ADS-B

ICAO ID
A835AF

CALLSIGN
N628TS

ALTITUDE
22,100 ft.

SPEED
395 kn.

HEADING
344°

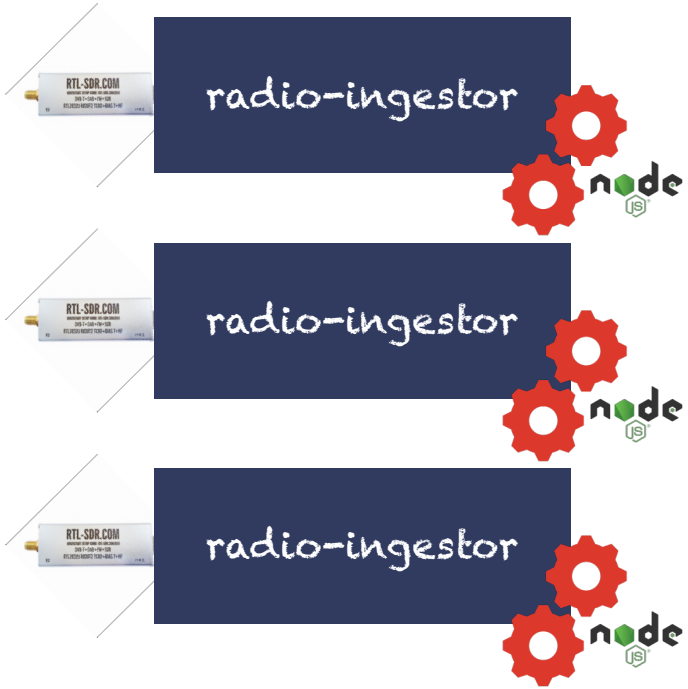
LOCATION
39.963°N
83.205°W

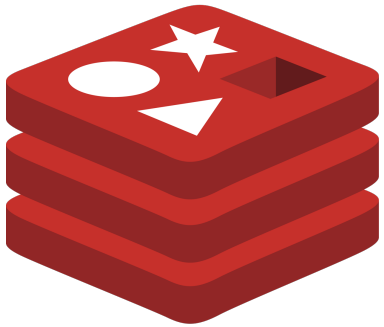
Tot: 15 Vis: 15 RSSI: Max -25.3+ Mean -32.4 Min -34.7- MaxD: 0.0nm+   /

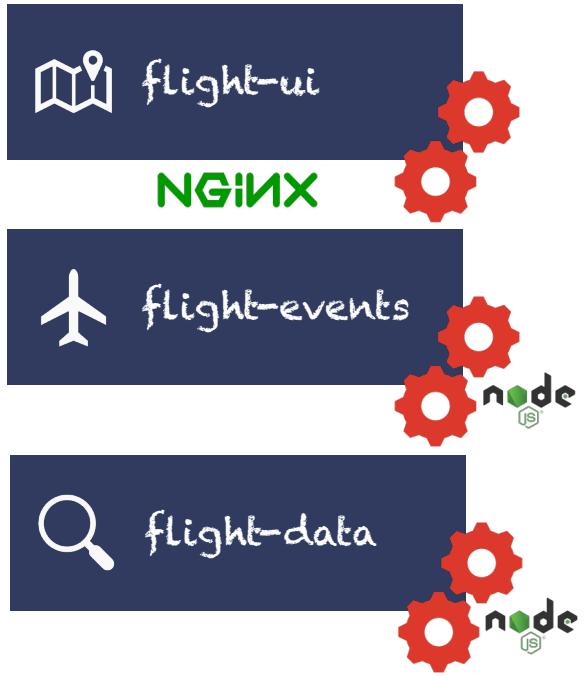
Hex	Mode	Sqwk	Flight	Alt	Spd	Hdg	Lat	Long	RSSI	Msgs	Ti
A446F1	A2	1640	LXJ375	22100	395	344	39.963	-83.205	-31.4	347	04
A296F4	A2	6746	N266TD	5575	189	273	39.994	-82.956	-30.5	123	30
A87108	A2	6702	PAC369	30650	523	051	40.384	-83.412	-33.3	286	00
A04E60	A2		RPA4747	30000	394	271	40.285	-82.991	-32.4	265	29
A655F1	A2	5612	RVJ55160	6075	296	098	40.097	-82.960	-25.3+	454	08
AAA244	A2	1040	UAL296	31000	494	101	40.369	-83.046	-31.5	1026	12
AC0417	A2	4063	ASH6330	9725	268	272	40.009	-83.261	-34.0	410	22
A2587B	A2	1535	XSR250	39000	534	059	40.560	-82.938	-33.6	688	52
AB2AE1	A2	2231		35025	575	059	40.676	-83.134	-34.7-	346	20
A8A70F	A0		DABJET55	36025	347	264	39.756	-83.036	-32.9	51	31
AA1A1F	A2		ABX3185	31000					-33.5	571	43
ABBBC7	A2		ASA1108	31025	517	098	40.238	-82.689	-32.0	885	44
ADA94D	A2	4164	N98FE	40000	348	239	39.852	-83.509	-34.1	2350	00
A575D3	A2		JRT51	43000	399	209	39.947	-83.783	-33.1	852	15
ACBCFC	A2		N92MZ	20000	210	281	40.257	-83.377	-33.7	1632	26

DEMO

redis

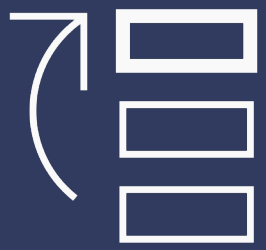

RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node




flight-ui
NGINX
flight-events
node
flight-data
node

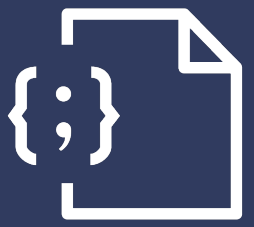

api-gateway
NGINX


radio-aggregator
node



STREAMS

- An event stream contained in a key
- Events have ID in format 12345-0
- Events have key-value data
- Included since Redis 5.0



REDISJSON

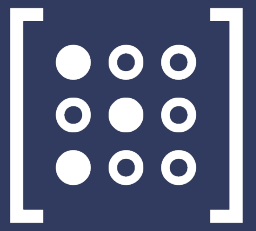
- A JSON document in a key in Redis
- Commands to modified & query
- Source available
- Included with Redis Stack

REDISEARCH

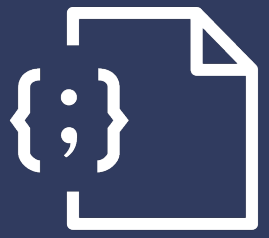
- Indexing & full-text search for Redis
- Searches & indexes Hashes
- Searches & indexes JSON
- Included with Redis Stack

redis stack

REDIS BLOOM



REDIS GRAPH

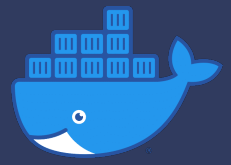


REDIS JSON

REDIS SEARCH

REDIS TIMESERIES

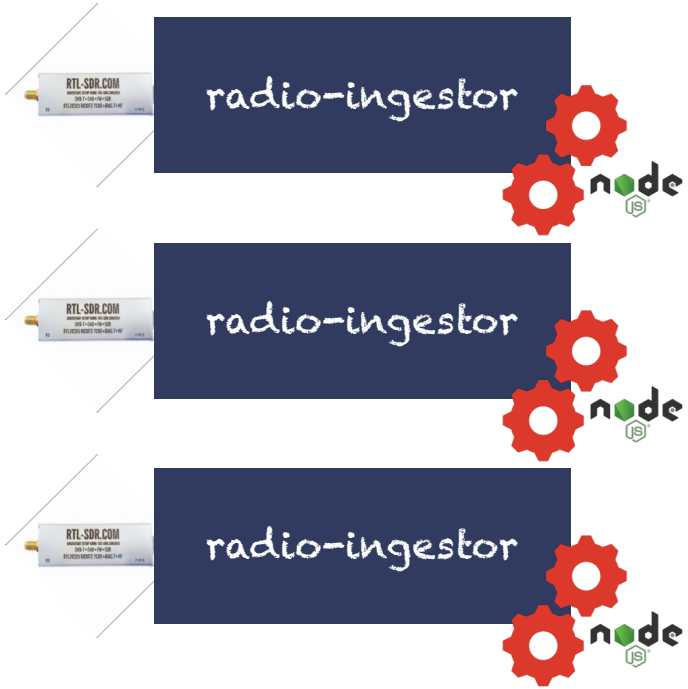
redis stack

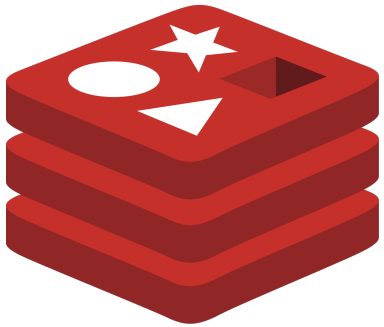


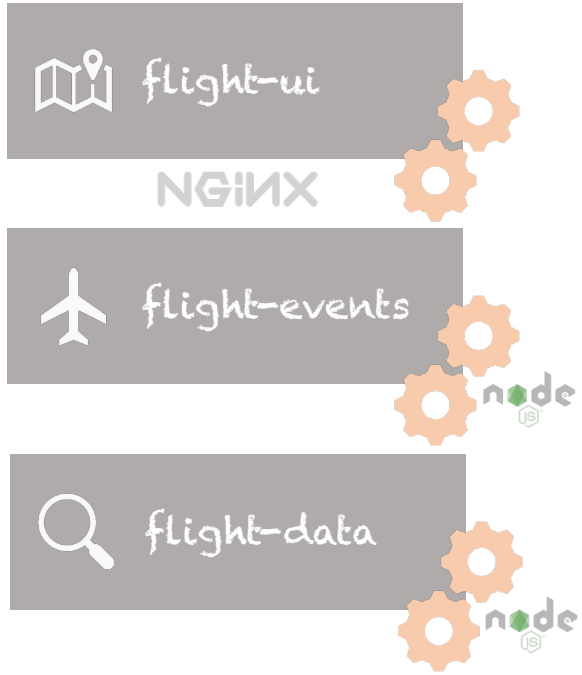
```
$ docker run -d --name redis-stack-server -p 6379:6379 \
    redis/redis-stack-server:latest
```

```
$ brew tap redis-stack/redis-stack
$ brew install redis-stack
```

```
$ open https://redis.io/docs/stack
$ open https://redis.com/try-free
```


RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node

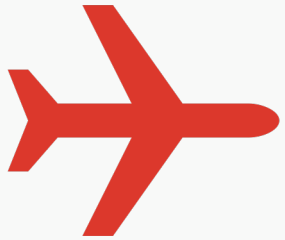



flight-ui
NGINX
flight-events
node
flight-data
node


api-gateway
NGINX


radio-aggregator
node

CONNECTING TO THINGS



```
import * as sbs1 from 'sbs1'
import * as redis from 'redis'

// connect to SBS1 source and await messages
const sbs1Client = sbs1.createClient({
  host: SBS1_HOST, port: SBS1_PORT })

// create a Redis client
const redisClient = redis.createClient({
  socket: { host: REDIS_HOST, port: REDIS_PORT },
  password: REDIS_PASSWORD
})

// catch and log any Redis errors
redisClient.on('error', (err) => console.log('Redis Client Error', err))

// connect to Redis
await redisClient.connect()
```

REGISTER THE INGESTOR

```
// add the stream key to a set for later aggregation
redisClient.sAdd(INGESTOR_STREAM_KEYS_KEY, INGESTOR_STREAM_KEY)
```

radios radio:krk

```
redis.cloud:6379> SADD radios radio:krk
(integer) 1
redis.cloud:6379> SADD radios radio:ema
(integer) 1
redis.cloud:6379> SADD radios radio:oak
(integer) 1
redis.cloud:6379> SADD radios radio:cmh
(integer) 1
```

```
redis.cloud:6379> SMEMBERS radios
1) "radio:oak"
2) "radio:cmh"
3) "radio:ema"
4) "radio:krk"
```

WAIT FOR MESSAGES

```
// connect to SBS1 source and await messages
sbs1Client.on('message', msg => {

  ...process the message...

})
```

PROCESSING THE MESSAGE

```
// add fields that are always in the message
const event = {
  icaoId: msg.hex_ident,
  type: toTransmissionType(msg.transmission_type),
  generatedDateTime: toEpochMilliseconds(msg.generated_date, msg.generated_time).toString(),
  loggedDateTime: toEpochMilliseconds(msg.logged_date, msg.logged_time).toString()
}

// add fields that might be in the message
if (msg.callsign !== null) event.callsign = msg.callsign.trim()
if (msg.altitude !== null) event.altitude = msg.altitude.toString()
if (msg.lat !== null) event.latitude = msg.lat.toString()
if (msg.lon !== null) event.longitude = msg.lon.toString()
if (msg.ground_speed !== null) event.velocity = msg.ground_speed.toString()
if (msg.track !== null) event.heading = msg.track.toString()
if (msg.vertical_rate !== null) event.climb = msg.vertical_rate.toString()
if (msg.is_on_ground !== null) event.onGround = msg.is_on_ground.toString()
```

ADDING THE EVENT

```
// add the event to the stream
redisClient.xAdd(INGESTOR_STREAM_KEY, '*', event)
```

radio:krk

```
redis.cloud:6379> XADD radio:krk * icaoId A835AF callsign N628TS ...
"1655410429863-0"
```

```
redis.cloud:6379> XRANGE radio:krk - +
1) 1) "1655410429863-0"
   2)  1) "icaoId"
       2) "A835AF"
       3) "callsign"
       4) "N628TS"
       5) ...
```

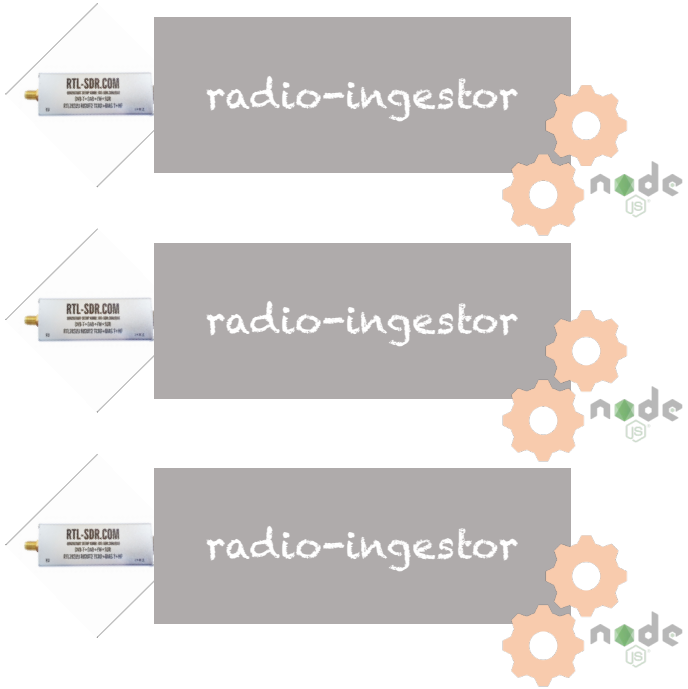
TRIMMING OLD EVENTS

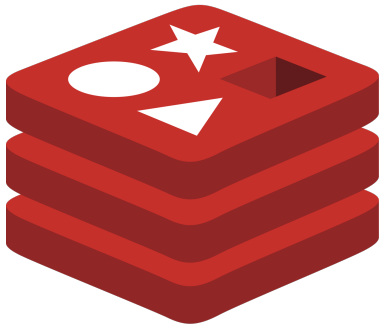
```
// find oldest event id to keep
const oldestEventId = new Date().getTime() - INGESTOR_STREAM_LIFETIME * 1000

// add the event to the stream and expire old events
redisClient.xAdd(INGESTOR_STREAM_KEY, '*', event, {
  TRIM: {
    strategy: 'MINID',
    threshold: oldestEventId
  }
})
```

```
redis.cloud:6379> XADD radio:krk * icaoId A835AF ... MINID 1655410429000-0
"1655411024042-0"
```

DEMO

redis


RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node




flight-ui
NGINX
flight-events
node
flight-data
node


api-gateway
NGINX


radio-aggregator
node

FIND THE STREAMS

```
// get the keys for all the ingestor streams from Redis
const streamKeys = await redisClient.sMembers(INGESTOR_STREAM_KEYS_KEY)
```

radios

```
redis.cloud:6379> SMEMBERS radios
1) "radio:oak"
2) "radio:cmh"
3) "radio:ema"
4) "radio:krk"
```

STARTING NOW

```
// build up the starting event IDs to look at... starting with right now
const startId = `${new Date().getTime()}-0`
const currentKeysAndIds = streamKeys.map(key => ({ key, id: startId }))

// read streams forever
while (true) {

  // wait at most a second for at most 10 events per stream key
  const results = await redisClient.xRead(currentKeysAndIds, {
    BLOCK: 1000,
    COUNT: 10
  }) ?? []

  ...process the events...

}
```

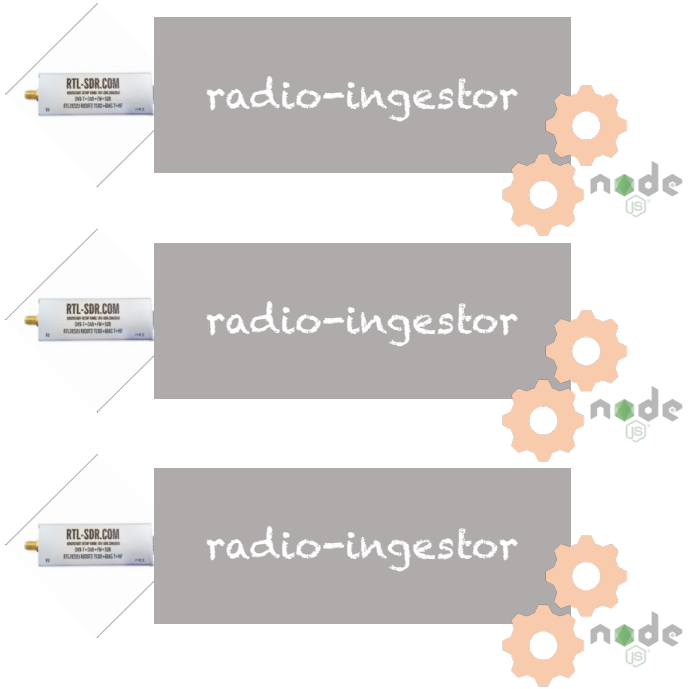
```
redis.cloud:6379> XREAD COUNT 10 BLOCK 1000 STREAMS \
    radio:krk radio:cmh radio:oak radio:ema \
    01655410429863-0 01655410429863-0 01655410429863-0 01655410429863-0
```

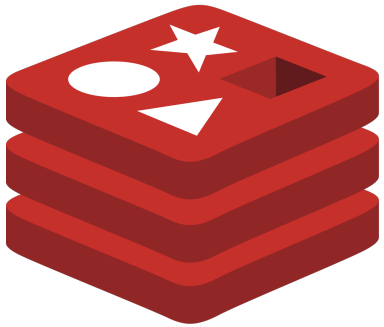
STARTING NOW

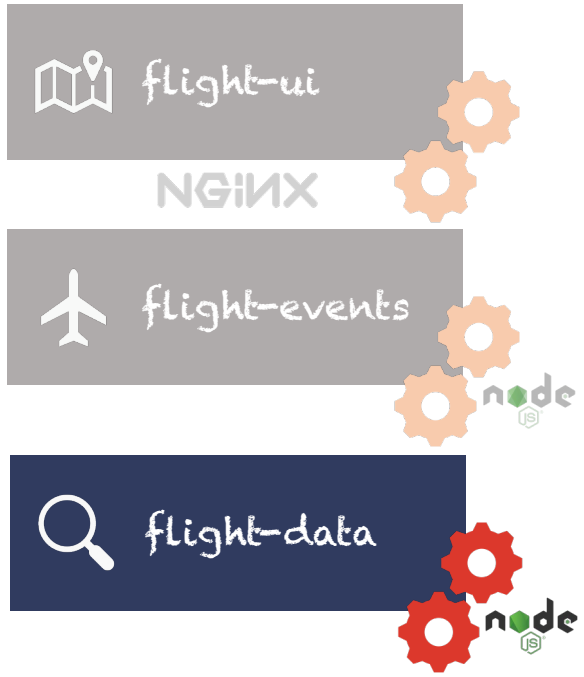
```
redis.cloud:6379> XREAD COUNT 10 BLOCK 1000 STREAMS \
    radio:krk radio:cmh radio:oak radio:ema \
    01655410429863-0 01655410429863-0 01655410429863-0 01655410429863-0
1) 1) "radio:krk"
   2) 1) 1) "1655410429863-0"
         2)  1) "icaoId"
             2) "A835AF"
             3) "callsign"
             4) "N628TS"
             5) ...
      2) 1) "1655411024042-0"
         2) 1) "icaoId"
            2) "A835AF"
            3) ...
2) 1) "radio:cmh"
   2) 1) 1) "1655480224410-0"
         2) 1) "icaoId"
            2) "A835AF"
            3) ...
```

PROCESSING THE EVENTS

```
// we can get a result for each ingestor stream (or none at all)
results.forEach(result => {

  // pull the values for the event out of the result
  const { name: streamKey, messages } = result
  const [ { id, message } ] = messages
  const event = { radio: streamKey, ...message }

  // update the current id with the most recently found id
  const index = currentKeysAndIds.findIndex(current => current.key === streamKey)
  currentKeysAndIds[index].id = id

  ...add the event to the all stream...

}
```


RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node




flight-ui
NGINX
flight-events
node
flight-data
node


api-gateway
NGINX


radio-aggregator
node


redisōM

SETTING UP REDIS OM


```
// create a Redis OM connection
export const omClient = await
  new Client().use(redisClient)

// define the Entity
class Flight extends Entity {}
```

```
// define the Schema
const flightSchema = new Schema(Flight, {
  icaoId: { type: 'string' },
  dateTime: { type: 'date' },
  radio: { type: 'string' },
  callsign: { type: 'string' },
  altitude: { type: 'number' },
  latitude: { type: 'number' },
  longitude: { type: 'number' },
  location: { type: 'point' },
  velocity: { type: 'number' },
  heading: { type: 'number' },
  climb: { type: 'number' },
  onGround: { type: 'boolean' }
})
```

```
// create the Repository and create the index
const flightRepository = omClient.fetchRepository(flightSchema)
await flightRepository.createIndex()
```

STREAMS TO JSON WITH REDIS OM

```
// fetch the flight
let flight = await flightRepository.search()
  .where('icaoId').equals(event.icaoId)
    .return.first()

// if we get nothing, we need to create it
flight = flight ?? flightRepository.createEntity()

// set the properties on the entity
flight.icaoId = event.icaoId
flight.dateTime = Number(event.loggedDateTime)
flight.radio = event.radio

...set other properties...

// save the flight
await flightRepository.save(flight)
```

A SIMPLE QUERY WITH EXPRESS

```
router.get('/flights', async (req, res) => {
  const flights = await flightRepository.search().return.all()
  res.send(flights)
})

router.get('/flight/icao/:id', async (req, res) => {
  const flight = await flightRepository.search()
    .where('icaoId').equals(req.params.id)
      .return.first()
  res.send(flight)
})

router.get('/flight/callsign/:callsign', async (req, res) => {
  const flight = await flightRepository.search()
    .where('callsign').equals(req.params.callsign)
      .return.first()
  res.send(flight)
})
```

express

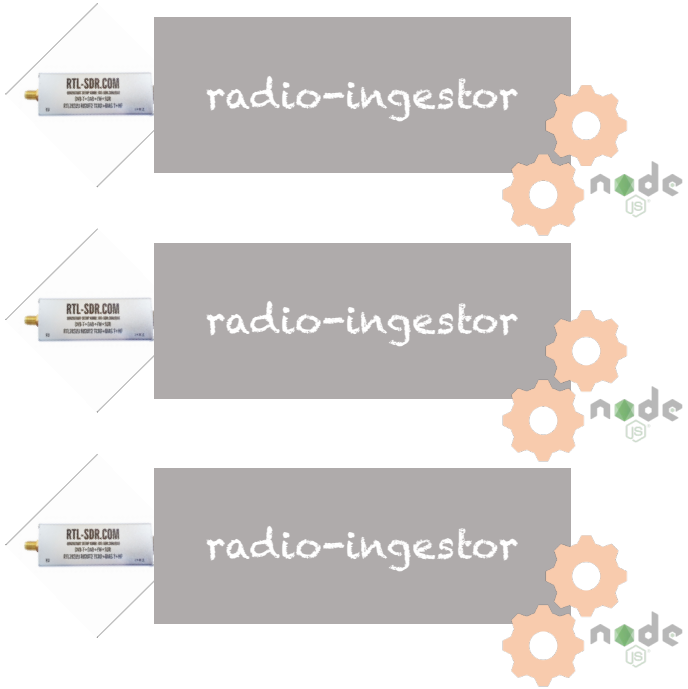
redisōM

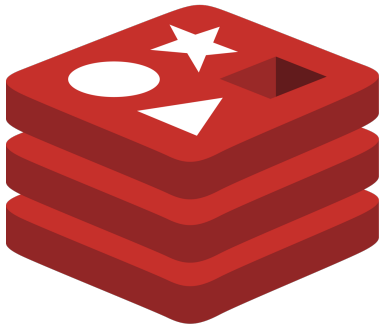
MORE INTERESTING QUERIES

```
router.get('/by-radio/:radio', async (req, res) => {
  const flights = await flightRepository.search()
    .where('radio').equals(req.params.radio)
      .return.all()
  res.send(flights)
})

router.get('/below/:altitude', async (req, res) => {
  const flights = await flightRepository.search()
    .where('altitude').lte(Number(req.params.altitude))
      .return.all()
  res.send(flights)
})

router.get('/heading/north', async (req, res) => {
  const flights = await flightRepository.search()
    .where('heading').between(0, 45)
      .or('heading').between(315, 360)
        .return.all()

  res.send(flights)
})
```

express

redisōM


RTL-SDR.COM
radio-ingestor
RTL-SDR.COM
radio-ingestor
RTL-SDR.COM
radio-ingestor

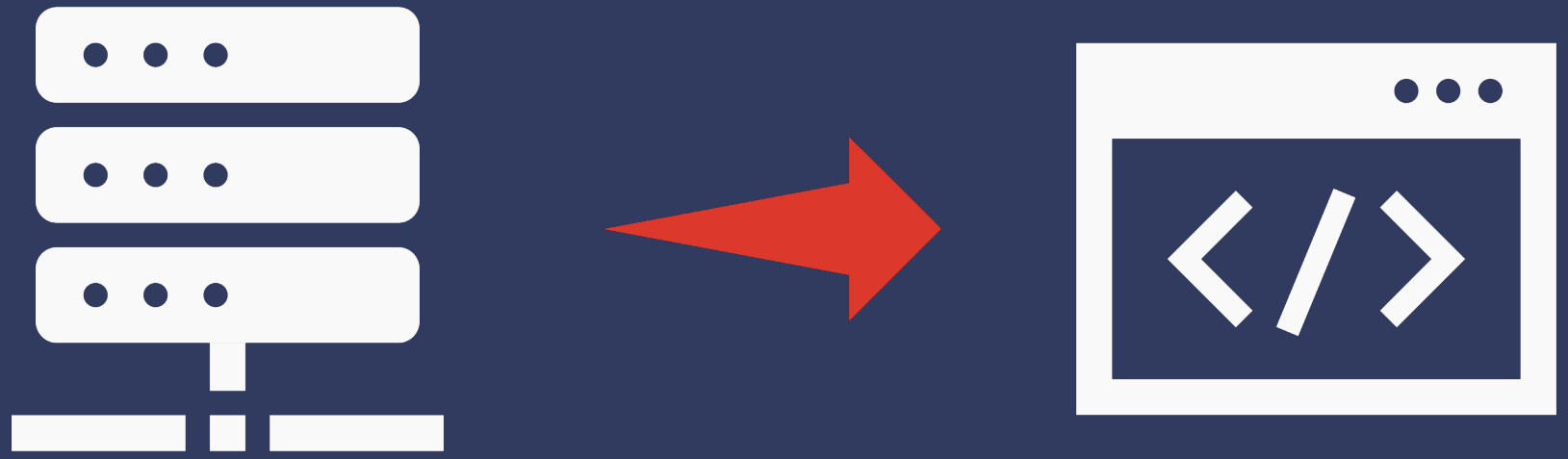



flight-ui
NGINX
flight-events
flight-data


api-gateway
NGINX


radio-aggregator

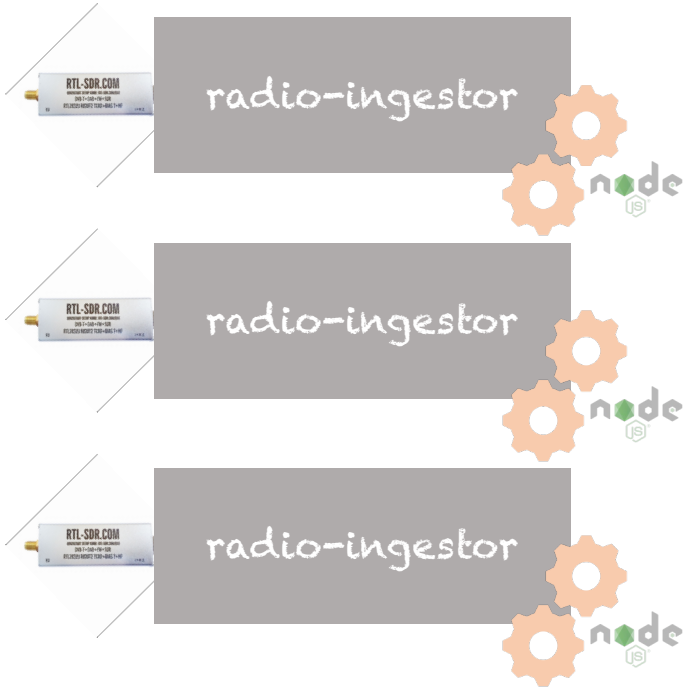
SERVER-SENT EVENTS

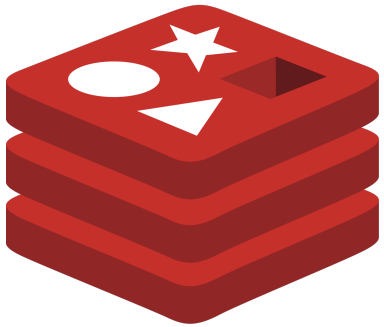


https://simonprickett.dev/a-look-at-server-sent-events/

PUBLISHING SERVER-SENT EVENTS

```
// set up server sent events for all aircraft
app.get('/events/flights/all', async (req, res) => {

  // set up headers for server sent events
  res.writeHead(200, {
    'Connection': 'keep-alive',
    'Content-Type': 'text/event-stream',
    'Cache-Control': 'no-cache'
  })

  // process events as they arrive
  let currentId = '$'
  while (true) {
    const result = await redisClient.xRead(...)
    const { id, message } = result[0].messages[0]
    currentId = id

    // set the server-sent event
    res.write(`id: ${id}\n`)
    res.write(`data: ${JSON.stringify({ ...message })}\n`)
    res.write(`\n`)
  }
})
```


RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node
RTL-SDR.COM
radio-ingestor
node




flight-ui
NGINX
flight-events
node
flight-data
node


api-gateway
NGINX

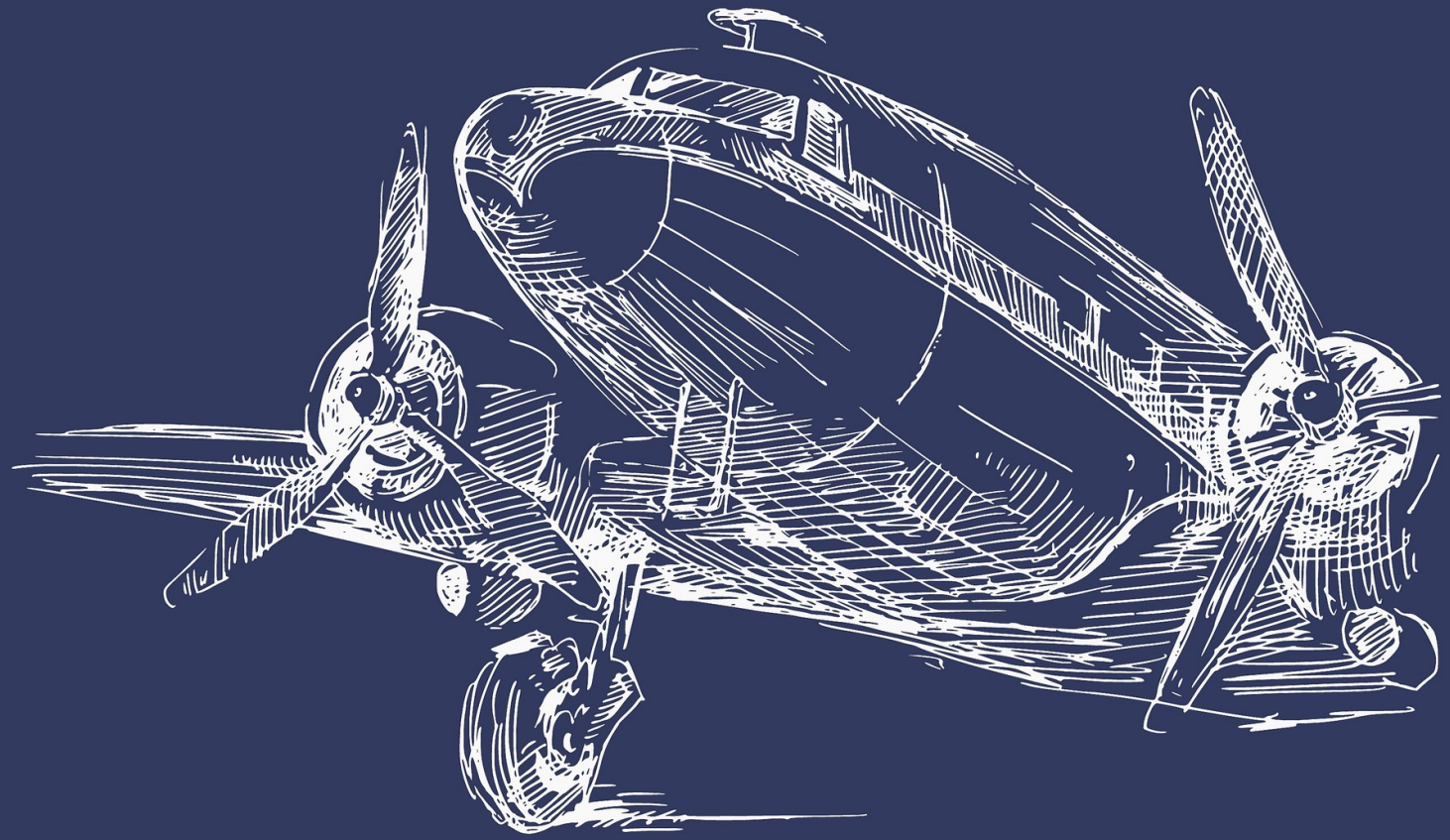

radio-aggregator
node

MAKING A MAP

RECEIVING SERVER-SENT EVENTS

```
// set up our event source and handle the events
const eventSource = new EventSource('/events/flights/all')
eventSource.onmessage = event => {

  // parse data out of the event
  const {
    icaoId, callsign, radio,
    latitude, longitude, altitude,
    heading, velocity, climb } = JSON.parse(event.data)

  ...do stuff with the data...

}
```

DEMO



redis

QUESTIONS?

redis

RESOURCES

Redis Stack
https://redis.io/docs/stack/

Redis Cloud
https://redis.com/try-free/

Redis OM for Node.js
https://github.com/redis/redis-om-node

AIS Trackers
https://www.vesselfinder.com/
https://www.marinetraffic.com/

ADS-B Trackers
https://flightaware.com/
https://globe.adsbexchange.com/

SDR Software
https://airspy.com/download/
https://gqrx.dk/
https://cubicsdr.com/
https://www.sdrangel.org/

dump1090
https://github.com/antirez/dump1090

Server-Sent Even ts
https://simonprickett.dev/a-look-at-server-sent-events/

REDIS DISCORD SERVER

https://discord.gg/redis

REDIS UNIVERSITY

https://university.redis.com/

REDIS STACK

https://redis.io/docs/stack/

SLIDES, CODE, AND WHATNOT

https://github.com/guyroyse/tracking-aircraft

Guy Royse

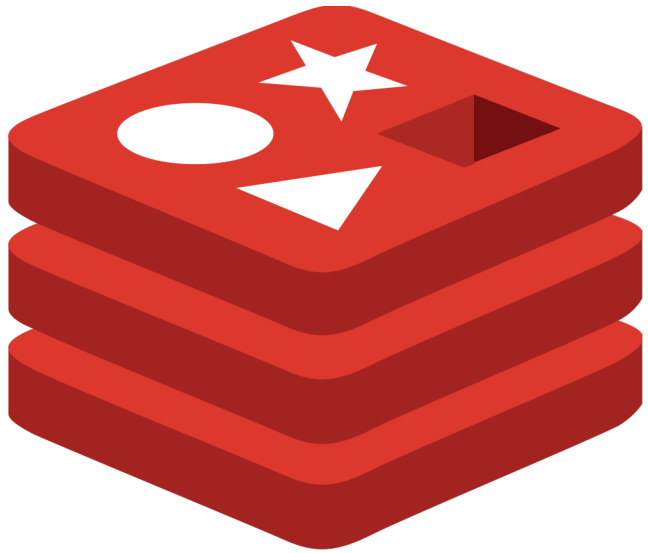
Developer Advocate

redis

@guyroyse

github.com/guyroyse

guy.dev



redis