General principles for constructing an internet access point using a homemade parabolic reflector, a 4G modem, and a feed antenna.

Dmitriy Makhnovskiy, Alexander Makhnovskiy, and Ludmila Makhnovskaya

Gardening plots 32 & 62, Non-commercial Gardening Association (NGA) “Rogachevo”, Trekhsvyatskoye Village, Dmitrovskiy District, Moscow Region, Russia

Technical consultation: dmitriy.makhnovskiy@gmail.com (Dmitriy)

Open access project repository on GitHub: https://github.com/DYK-Team/Parabolic_reflector

Use hyperlinks: by clicking on the underlined text, you will follow the link to a webpage

Contents

Goals and objectives of the project

The initial goal of this project is to provide a more or less stable 4G internet connection at a countryside house. There is no universal solution in this case, as signal reception conditions vary significantly depending on the terrain and can change even within small areas due to various obstacles, such as buildings, elevations, depressions, or forests. These factors can create a complex interference pattern that affects signal strength. The first step is to select the mobile carrier with the best 4G coverage in your area. It is also helpful to gather insights from neighbours who have already attempted to set up internet access and ask about the equipment they are using. However, even at this stage, you may encounter conflicting reviews, which can introduce uncertainty. In such a situation, it is best to start with technically sound and budget-friendly solutions.

Our project is intended for those who, driven by curiosity and with some free time, seek to acquire new practical knowledge and skills. The core idea of our approach involves creating a large-diameter homemade parabolic reflector, which is further equipped with a MIMO or SISO feed antenna for connection to a 4G modem. We successfully installed such a reflector in the attic, making this solution particularly cost-effective. In the future, the project may evolve towards the development of a more advanced receiving station with a freestanding mast, atop which a parabolic reflector in a spherical protective casing would be mounted. All the necessary equipment would be housed inside the casing. Naturally, this would be a much more expensive system, but its creation would be practically justified if it provides internet access to several gardening plots. *The educational aspect of the project, both at its current stage and in future developments, plays a key role for us and serves as an important source of motivation. In the context of rapid advancements in telecommunications and computer technologies, we are keen to gain a deeper understanding of these fields.*

In the presented report, which primarily covers the structural aspects of the project, we review and elaborate on the following methods:

- Algorithm for cutting a parabolic reflector from individual segments (petals).
- Algorithm for cutting the protective spherical casing for the parabolic reflector.
- Implementation of cutting algorithms in Python and JavaScript.
- Technology for manufacturing the segments from readily available materials.
- Assembly of the cut surfaces.
- Installation of the parabolic reflector in the attic of the house (without a casing).
- Determining the optimal azimuth orientation of the reflector based on the geolocation of the house and the nearest cellular base station.
- Selecting a 4G modem and irradiator.
- Key principles for constructing a mast station with a parabolic reflector for enhanced signal reception.
- Key principles of creating electronic files for laser cutting petals.

The fabrication of large-diameter parabolic reflectors can be particularly relevant for remote areas with poor signal reception. Purchasing a reflector with a diameter of one meter or more is quite difficult, and may even be impossible. Even if such a reflector is found, transporting it (if it is non-collapsible) would be extremely challenging.

The project repository, including the report, Python code, and a technical photo gallery, is available in open access on the GitHub platform. It can be downloaded in its entirety. The development of the feed antenna, the electromechanical system for reflector orientation, and other electronics will be discussed in the next report, which will also be published in open access.

Cutting petals for paraboloid fabrication

The geometric parameters required for calculating the segments from which the paraboloid of rotation will be constructed are shown in Fig. 1. The simplest equation of a parabola in a rectangular XZ coordinate system is as follows:

$$z(x) = ax^2 \quad (1)$$

where a is a positive or negative constant determines the aperture of the parabola and the direction of its concavity. The paraboloid is generated by rotating this curve around the Z-axis. Thus, the parabola forms the line that generates the surface. The segments (petals) will be formed by cutting the surface along parabola lines spaced at equal angular intervals around the axis of rotation.

For each radius r, measured from the axis of rotation, there corresponds a specific length of the parabola $l(r)$, calculated using the following tabular integral:

$$l(r) = \int_0^r \sqrt{1 + (z'(s))^2}ds = \int_0^r \sqrt{1 + 4a^2s^2}ds = \frac{1}{2a}\int_0^{2ar} \sqrt{1 + s^2}ds = \frac{1}{2}r\sqrt{1 + 4a^2r^2} + \frac{1}{4a}\ln\left(2ar + \sqrt{1 + 4a^2r^2}\right) \quad (2)$$

For a total reflector diameter $D_r = 2R$, we obtain the length L of each segment as follows:

$$L = l(R) = \frac{1}{2}R\sqrt{1 + 4a^2R^2} + \frac{1}{4a}\ln\left(2aR + \sqrt{1 + 4a^2R^2}\right) \quad (3)$$

To determine the profile of the petal, we will need to find the value of r for a given l along the parabola, which will be achieved by numerically solving the nonlinear equation using the bisection method:

$$\forall l \in [0, L], \exists r \in [0, R]: \frac{1}{2}r\sqrt{1 + 4a^2r^2} + \frac{1}{4a}\ln\left(2ar + \sqrt{1 + 4a^2r^2}\right) - l = 0 \quad (4)$$

The algorithm for solving this equation is provided below.

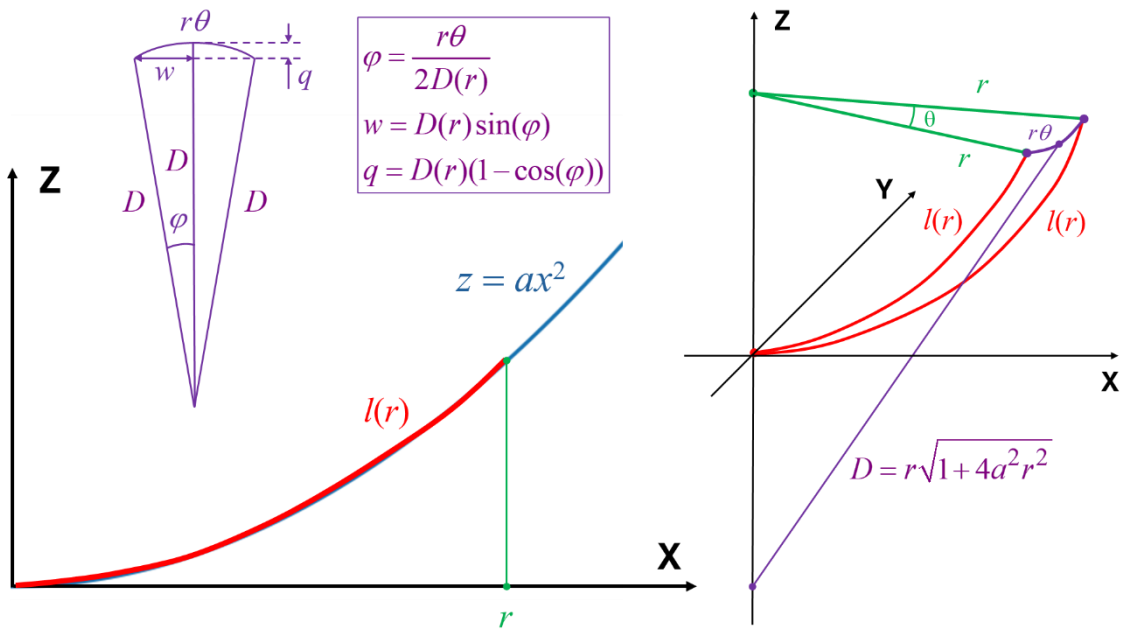


Fig. 1. Geometric parameters for calculating the petals from which the paraboloid will be constructed.

To understand the general principles of cutting surfaces of rotation into petals, let us consider two simple shapes in Fig. 2 – a cone and a cylinder. When dividing a surface of rotation into N identical petals, the angular width of each segment is $\theta = 2\pi/N$, and it remains constant along the length of the segment. To construct the profile of the petal, it is necessary to know its total width $2w$ for the values $s = l - q$ along the meridian of the petal at the current length l, where q is the height of the curvature of the petal, as shown in Fig. 2. In the case of a cone, the width of the petal is directly proportional to its length:

$$\varphi = \frac{r\theta}{2l} = const \quad (5)$$

$$w = l\sin(\varphi) = s\tan(\varphi) \quad (6)$$

$$q = l(1-\cos(\varphi)) \quad (7)$$

Thus, the profile of the petal consists of an equilateral triangle with a curvature at the apex, where the radius of curvature is equal to the maximum length of the petal $l = L$ along the cone surface. In the case of a cylinder, the width of the petal remains constant, and the radius of curvature at the ends of the segment is infinite. Therefore, its profile will be a rectangle. Both of these cases are unified by introducing the parameter $D(l)$, equal to the length of the tangent segment connecting the apex of the petal to the axis of rotation. The radius of curvature of the closing arc of the petal will be equal to D. In the case of a cone, this segment lies along the slanted line that forms the surface, and, consequently, $D = l$. In the case of a cylinder, this segment also lies along the generating line; however, its length becomes infinite, as the line does not intersect the axis of rotation. An infinite radius of curvature implies a straight line, resulting in a rectangular petal profile. The value of D can be calculated for specific surfaces of rotation. We are particularly interested in the paraboloid of rotation (see Fig. 1) and the sphere.

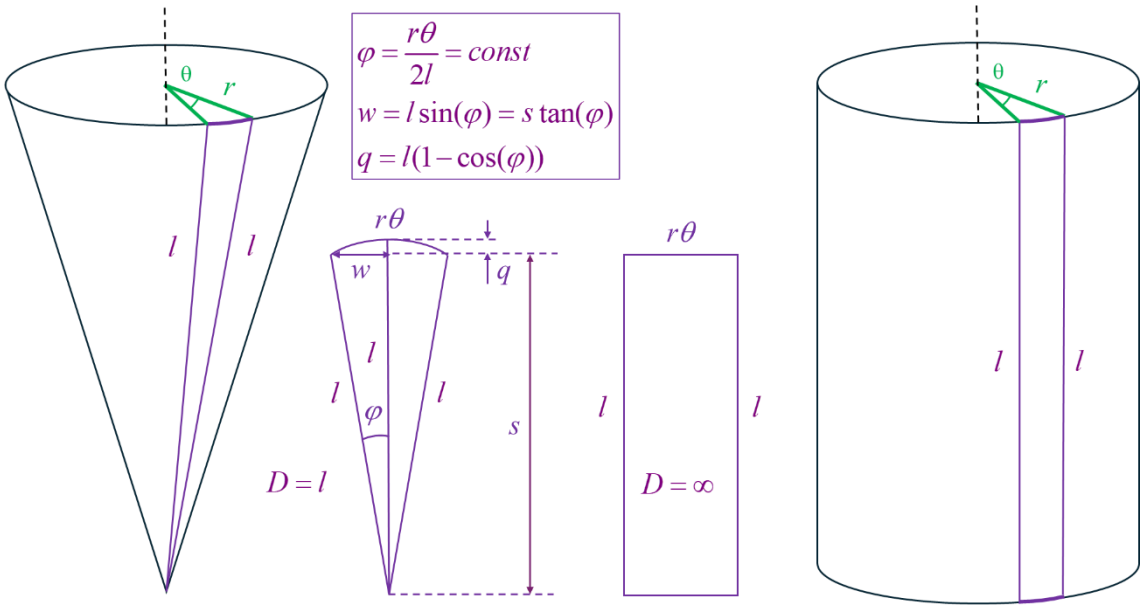


Fig. 2. Cutting of the cone and cylinder, as the simplest examples.

Based on the cutting examples of the cone and cylinder discussed above, we can formulate an algorithm for constructing the profile of a petal of a paraboloid of rotation:

1. Calculate the angular width of the petal $\theta = 2\pi/N$, where N is the number of petals.
2. For the given R and a calculate the total length of the petal L using eq. (3).
3. Calculate the array of petal lengths $l_i = iL/(M-1)$, $i = \overline{0, M-1}$, where M is the enough number of divisions, which determine the accuracy of the profile. Numbering starts from $i = 0$, as is customary in programming.
4. Calculate the array of radii r_i, corresponding l_i, solving the non-linear equation (4) for $0 < i < M-1$. In this case, $r_0 = 0$ and $r_{M-1} = R$.
5. Calculate the array of tangent segments $D_i = r_i\sqrt{1+4a^2r_i^2}$, $i = \overline{0, M-1}$.
6. Calculate the array of opening angles of the petals $\varphi_i = \frac{r_i\theta}{2D_i} = \frac{\theta}{2\sqrt{1+4a^2r_i^2}}$, $i = \overline{0, M-1}$.
7. Calculate the array of displacements $q_i = D_i(1-\cos(\varphi_i))$, $i = \overline{0, M-1}$.
8. Calculate the array of meridional coordinates $s_i = l_i - q_i$, $i = \overline{0, M-1}$.
9. Calculate the array of widths $w_i = D_i\sin(\varphi_i)$, $i = \overline{0, M-1}$, measured from the corresponding meridional coordinates s_i in both directions.
10. To close the petal, connect the ends of the final widths w_{M-1} with the apex of the petal $s_M = L$ (additional meridional coordinate) along the arc of the radius of curvature $D_{M-1} = R\sqrt{1+4a^2R^2}$.

As the primary parameters of the paraboloid of rotation, we will use the maximum radius of the reflector R and its focus length f. As an exercise, let us prove the well-known formula for calculating the parameter of a parabola a through f:

$$a = 1/(4f) \tag{8}$$

For this, refer to Fig. 3, which clearly demonstrates the property of a paraboloid to reflect rays, parallel to its axis, toward a single point called the focus. The path of one such ray is shown by the red lines in the figure.

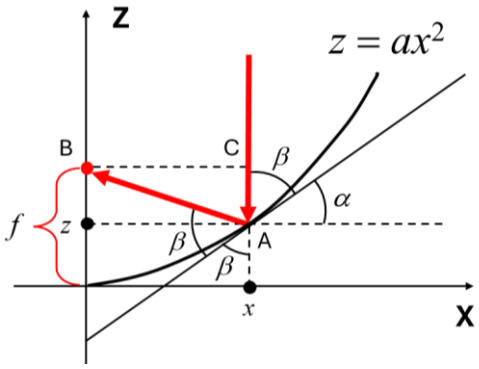


Fig. 3. Geometric parameters used for calculating the focal length f.

For the tangent angle α and the angle of incidence of the ray β, we obtain:

$$\tan(\alpha) = z'(x) = 2ax \tag{9}$$

$$\beta = \frac{\pi}{2} - \alpha \tag{10}$$

$$\tan(\beta) = \frac{1}{\tan(\alpha)} \tag{11}$$

Next, for the right angle triangle ABC, we have the following relationship (the angle of incidence is equal to the angle of reflection relative to the tangent line at the point where the ray strikes the surface):

$$\frac{x}{f-z} = \tan(\pi - 2\beta) = -\tan(2\beta) = -\frac{2\sin(\beta)\cos(\beta)}{\cos^2(\beta)-\sin^2(\beta)} = \frac{2\tan(\beta)}{\tan^2(\beta)-1} \tag{12}$$

Relation (12) remains valid for any position of point x, including when point z is above the focus. Triangle ABC was used only to derive the relation. Using equations (9), (11), and (12), we finally obtain:

$$\frac{x}{f-z} = \frac{2\tan(\alpha)}{1-\tan^2(\alpha)}$$

$$\frac{x}{f-ax^2} = \frac{4ax}{1-4a^2x^2}$$

$$\frac{1}{f-ax^2} = \frac{1}{\frac{1}{4a}-ax^2}$$

$$f = \frac{1}{4a} \tag{13}$$

Thus, we have proven that the focus indeed exists, as it does not depend on the point at which a parallel ray strikes the paraboloid.

The profile of the petal will be generated by the table of values (s_i, w_i) and the closing arc, as outlined in steps 8–10 of the algorithm. Below, we provide the Python code for calculating the profile. It can be downloaded from the project repository (file: Parabolic_reflector.py). The program requires two Python libraries: csv and numpy. Any IDE environment can be used for debugging and running the program. We prefer to work in the free version of PyCharm Community Edition, which makes it easy to download the necessary libraries.

```
#
# Algorithm for cutting the petals for a parabolic reflector
#
# Dmitriy Makhnovskiy, August 2024
#

import csv
import numpy as np

pi = np.pi  # pi-constant 3.1415....
c = 2.99792458e8  # speed of light in m/s

# Design parameters
units = 'mm'  # your length units: mm, cm, m (use small letters)
# For R and f use the same units: mm, cm, or m
R = 500.0  # dish radius
f = 500.0  # focus length
N = 10  # number of petals used for the parabolic reflector
M = 50  # number of points on the petal template for drawing its profile
eps = 1.0e-10  # calculation precision of the roots of the non-linear equation
# The minimum and maximum frequencies used for the communication, for example, those used for 4G LTE
f_min = 0.9  # minimum in GHz
f_max = 2.6  # maximum in GHz
k = 0.6  # reflector efficiency (0-1)

# Calculated parameters
f_min = f_min * 1.0e9  # frequency in Hz
f_max = f_max * 1.0e9  # frequency in Hz

lambda_max = c / f_min  # maximal wavelength
lambda_min = c / f_max  # minimal wavelength
if units == 'mm':
    radius = R / 1000.0
elif units == 'cm':
    radius = R / 100.0
else:
    radius = R

# Antenna gain range [Gain_min, Gain_max] dB for the given reflector efficiency k
Gain_min = 10.0 * np.log10(k * (2.0 * pi * radius / lambda_max)**2)
Gain_max = 10.0 * np.log10(k * (2.0 * pi * radius / lambda_min)**2)

a = 1.0 / (4.0 * f)  # parabola coefficient, z(x) = a*x^2
theta = 2.0 * pi / N  # angular width of the petal
value = np.sqrt(1.0 + 4.0 * a**2 * R**2)
L = R * value / 2.0 + np.log(2.0 * a * R + value) * f  # petal length

# Calculating the radius r from the length l along the parabola using the method of "dividing by half"
def rl(length):
    left = 0.0
    right = R
    r = (left + right) / 2.0  # initial middle point
    value = np.sqrt(1.0 + 4.0 * a ** 2 * r ** 2)
    value = r * value / 2.0 + np.log(2.0 * a * r + value) * f - length
    i = 1  # interation counter
    # Shifting the boundaries
    while np.abs(value) > eps and i < 100:
        if value < 0.0:
            left = r
```

```
        elif value > 0.0:
            right = r
        r = (left + right) / 2.0  # new middle point
        value = np.sqrt(1.0 + 4.0 * a ** 2 * r ** 2)
        value = r * value / 2.0 + np.log(2.0 * a * r + value) * f - length
        i = i + 1  # number of iterations
    return r

l = [0.0] * M  # points along the petal
l[M - 1] = L

r = [0.0] * M  # array of r corresponding to the array of l
r[M - 1] = R

D = [0.0] * M  # array of the tangent segments
D[M - 1] = R * np.sqrt(1.0 + 4.0 * a**2 * R**2)

eff = [0.0] * M  # array of the opening angles
eff[0] = theta / 2.0
eff[M - 1] = theta / (2.0 * np.sqrt(1.0 + 4.0 * a**2 * R**2))

q = [0.0] * M  # meridian displacements with respect to the l-points
q[M - 1] = D[M - 1] * (1.0 - np.cos(eff[M - 1]))

s = [0.0] * M  # array of the meridian coordinates
s[M - 1] = l[M - 1] - q[M - 1]

w = [0.0] * M  # array of the widths corresponding to the array of s
w[M - 1] = D[M - 1] * np.sin(eff[M - 1])

for i in range(1, M-1):
    length = i * L / (M - 1)
    l[i] = length
    r[i] = rl(length)
    D[i] = r[i] * np.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2)
    eff[i] = theta / (2.0 * np.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2))
    q[i] = D[i] * (1.0 - np.cos(eff[i]))
    s[i] = l[i] - q[i]
    w[i] = D[i] * np.sin(eff[i])

# Saving the profile data to a CSV file
data = np.column_stack((l, r, D, eff, q, s, w))
header = ['l', 'r', 'D', 'eff', 'q', 's', 'w']

with open('Parabolic_profile_data.csv', 'w', newline='') as csv_file:
    writer = csv.writer(csv_file)
    writer.writerow(header)
    writer.writerows(data)

# Saving the design parameters to a txt file
Dish_radius = 'Dish radius = ' + str(R) + ' ' + units + '\n'
Dish_height = 'Dish height = ' + str(a * R**2) + ' ' + units + '\n'
Focus_length = 'Focus length of the reflector= ' + str(f) + ' ' + units + '\n'
Maximum_length = 'Petal length = ' + str(L) + ' ' + units + '\n'
N_number = 'Number of petals = ' + str(N) + '\n'
Petal_radius = 'Petal radius of curvature = ' + str(D[M - 1]) + ' ' + units +  '\n'
Frequency_min = 'Minimum frequency = ' + str(f_min / 1.0e9) + ' GHz' + '\n'
Frequency_max = 'Maximum frequency = ' + str(f_max / 1.0e9) + ' GHz' + '\n'
Wavelength_min = 'Minimum wavelength = ' + str(lambda_min) + ' m' + '\n'
Wavelength_max = 'Maximum wavelength = ' + str(lambda_max) + ' m' + '\n'
Reflector_eff = 'Reflector efficiency = ' + str(k) + '\n'
Antenna_Gain_min = 'Minimum antenna gain = ' + str(Gain_min) + ' dB' + '\n'
Antenna_Gain_max = 'Maximum antenna gain = ' + str(Gain_max) + ' dB' + '\n'

design_parameters = open('Design_parameters.txt', 'w')
design_parameters.write(Dish_radius)
design_parameters.write(Dish_height)
design_parameters.write(Focus_length)
design_parameters.write(Maximum_length)
design_parameters.write(N_number)
design_parameters.write(Petal_radius)
design_parameters.write(Frequency_min)
```

```
132. design_parameters.write(Frequency_max)
133. design_parameters.write(Wavelength_min)
134. design_parameters.write(Wavelength_max)
135. design_parameters.write(Reflector_eff)
136. design_parameters.write(Antenna_Gain_min)
137. design_parameters.write(Antenna_Gain_max)
138. design_parameters.close()
```

The program is a console application and, therefore, does not have a graphical user interface (GUI). All design parameters must be entered directly into the program's code:

```
13. # Design parameters
14. units = 'mm'  # your length units: mm, cm, m (use small letters)
15. # For R and f use the same units: mm, cm, or m
16. R = 500.0  # dish radius
17. f = 500.0  # focus length
18. N = 10  # number of petals used for the parabolic reflector
19. M = 50  # number of points on the petal template for drawing its profile
20. eps = 1.0e-10  # calculation precision of the roots of the non-linear equation
21. # The minimum and maximum frequencies used for the communication, for example, those used for 4G LTE
22. f_min = 0.9  # minimum frequency in GHz
23. f_max = 2.6  # maximum frequency in GHz
24. k = 0.6  # reflector efficiency (0-1)
```

The bisection algorithm (or “method of dividing by half”) for solving the nonlinear equation (4) is implemented as the function rl(length). This function is called each time the current radius r needs to be found for the current meridional coordinate l. The accuracy of the root calculation is controlled by the absolute difference $\text{eps} = 10^{-10}$ in equation (4). The maximum number of iterations for finding the root is limited to 100, preventing potential “oscillation” around the root if the specified calculation accuracy is too high.

```
48. # Calculating the radius r from the length l along the parabola using the method of "dividing by half"
49. def rl(length):
50.     left = 0.0
51.     right = R
52.     r = (left + right) / 2.0  # initial middle point
53.     value = np.sqrt(1.0 + 4.0 * a ** 2 * r ** 2)
54.     value = r * value / 2.0 + np.log(2.0 * a * r + value) * f - length
55.     i = 1  # interation counter
56.     # Shifting the boundaries
57.     while np.abs(value) > eps and i < 100:
58.         if value < 0.0:
59.             left = r
60.         elif value > 0.0:
61.             right = r
62.         r = (left + right) / 2.0  # new middle point
63.         value = np.sqrt(1.0 + 4.0 * a ** 2 * r ** 2)
64.         value = r * value / 2.0 + np.log(2.0 * a * r + value) * f - length
65.         i = i + 1  # number of iterations
66.     return r
```

As a result of executing the program, a tabular file Parabolic_profile_data.csv is created (see Fig. 4), with columns containing all the discrete parameters calculated in the algorithm (“eff” stands for φ). For constructing the petal profile (s_i, w_i) only the last two columns will be needed s and w. The radius of curvature of the closing arc is equal to the last value in column D.

	l	r	D	eff	q	s	w
2	0	0	0	0.314159	0	0	0
3	10.61458	10.61438	10.61498	0.314142	0.519476	10.0951	3.28003
4	21.22916	21.22757	21.23235	0.314089	1.038721	20.19044	6.559727
5	31.84374	31.83836	31.85449	0.314	1.557504	30.28624	9.83876
6	42.45832	42.44558	42.4838	0.313877	2.075596	40.38272	13.1168
7	53.0729	53.04803	53.12262	0.313718	2.59277	50.48013	16.3935
8	63.68748	63.64454	63.77331	0.313525	3.108799	60.57868	19.66856
9	74.30206	74.23393	74.43819	0.313297	3.623461	70.6786	22.94163
10	84.91664	84.81506	85.11958	0.313035	4.136537	80.7801	26.21239
11	95.53122	95.38677	95.81973	0.31274	4.647811	90.88341	29.48053
12	106.1458	105.9479	106.5409	0.312411	5.157071	100.9887	32.74573

Fig. 4. Structure of the tabular file Parabolic_profile_data.csv generated by the program.

The program also saves all design parameters in a text file Design_parameters.txt. An example of the contents of this file is shown below for a reflector with a radius of 500 mm and a focal length of 500 mm:

Dish radius = 500.0 mm
Dish height = 125.0 mm
Focus length of the reflector= 500.0 mm
Petal length = 520.1144097172755 mm)
Number of petals = 10
Petal radius of curvature = 559.0169943749474 mm
Minimum frequency = 0.9 GHz
Maximum frequency = 2.6 GHz
Minimum wavelength = 0.11530479153846154 m
Maximum wavelength = 0.3331027311111111 m
Reflector efficiency = 0.6
Minimum antenna gain = 17.27294608794706 dB
Maximum antenna gain = 26.487562858576922 dB

The petal profile calculated for these parameters is shown in Fig. 5. Although it is very close to a triangular shape (red straight lines), it has a slight outward bulge (“barrel-shaped”).

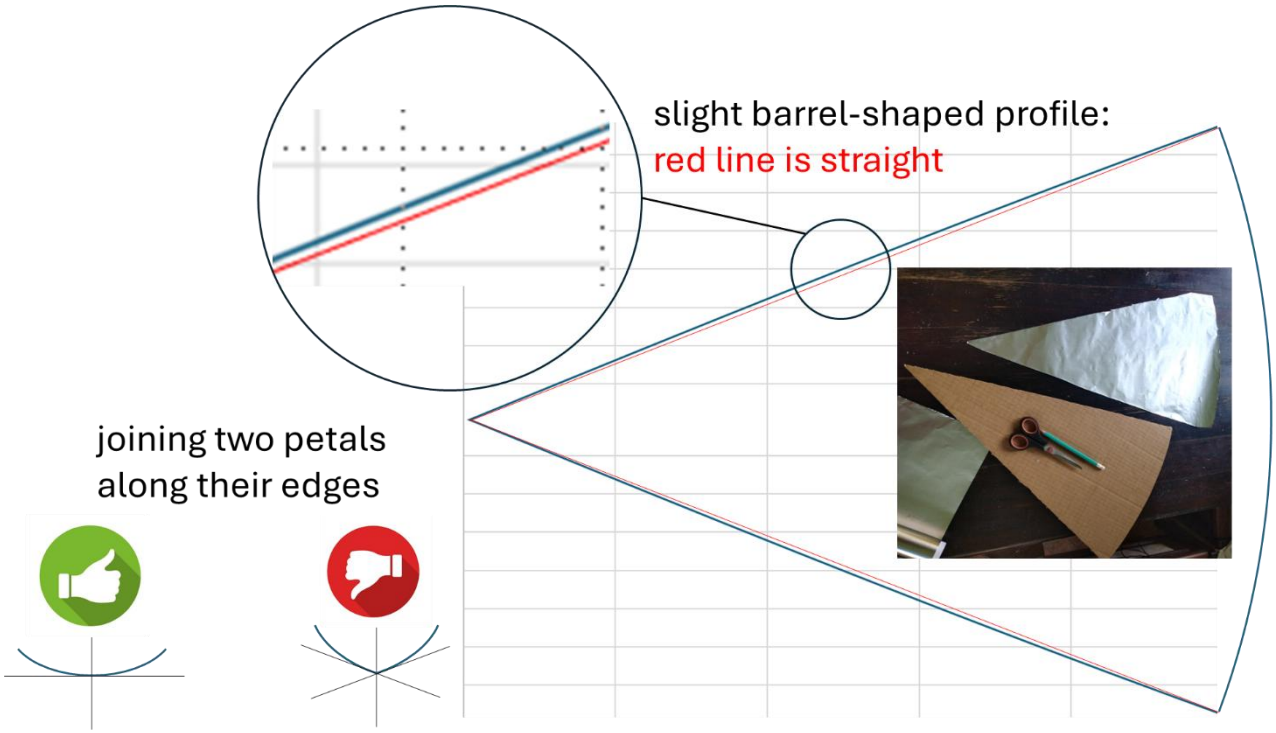


Fig. 5. Petal profile for a paraboloid with a radius of 500 mm and a focal length of 500 mm.

The accuracy of reproducing the parabolic surface is one of the key factors affecting the reflector's efficiency coefficient k (reflector efficiency), though not the only one. Therefore, it is crucial to cut the petals with the utmost precision, which poses significant challenges when cutting manually using a template. The petals must be joined precisely along the seams so that at every point along the seam, both petals share a common tangent plane, as shown in Fig. 5. Our first reflector prototype was assembled from petals cut out of packing cardboard. It is evident that we were unable to achieve high cutting accuracy. Furthermore, we used adhesive tape for assembly, which could not perfectly hold the petals in a single tangent plane along the seam, resulting in a slight misalignment. While the final shape turned out to be fairly acceptable, it is still far from ideal. The technological aspects of reflector fabrication are discussed in the next section.

Technology for manufacturing a parabolic reflector from petals

Our first reflector prototype was made from packing cardboard. This might seem amusing, but let us not forget that during World War II, fighter planes were made of plywood — as long as it flew and fired, it was effective! In our case, the result was similar: the shape of the reflector turned out to be fully functional. The main issues, as mentioned in the previous section, were the accuracy of cutting the template and the petals, as well as ensuring tangent alignment when gluing the seams. A detailed photo gallery of our experience is available in the project repository in the “Design gallery” folder.

The stages of cutting the petals are shown in Fig. 6. The template was drawn based on the calculated table of values (s_i, w_i), where the points s_i were first plotted along a straight line, and then for each of them, the corresponding widths w_i were marked perpendicular to the line on both sides. As long as the length of w_i allowed, a more precise calliper (to fractions of a millimetre) was used. For wider petals, a ruler had to be used, which is less accurate. Afterward, the ends of the perpendicular segments were connected with smooth lines, interpolating the profile of the petal. The closing arc of the petal was drawn using a wire as a radius (see Fig. 6). The remaining petals were cut out with a knife along the pencil outline created with the template. Crucially for subsequent assembly, the profile of the petal must be aligned along the internal corrugations of the cardboard (see Fig. 6). This ensures the petals retain elasticity, allowing them to better follow the parabolic curve and maintain the intended shape. The same template was used to mark the petals on 20-micron-thick kitchen aluminum foil, which was used to line the reflecting surface of the paraboloid.

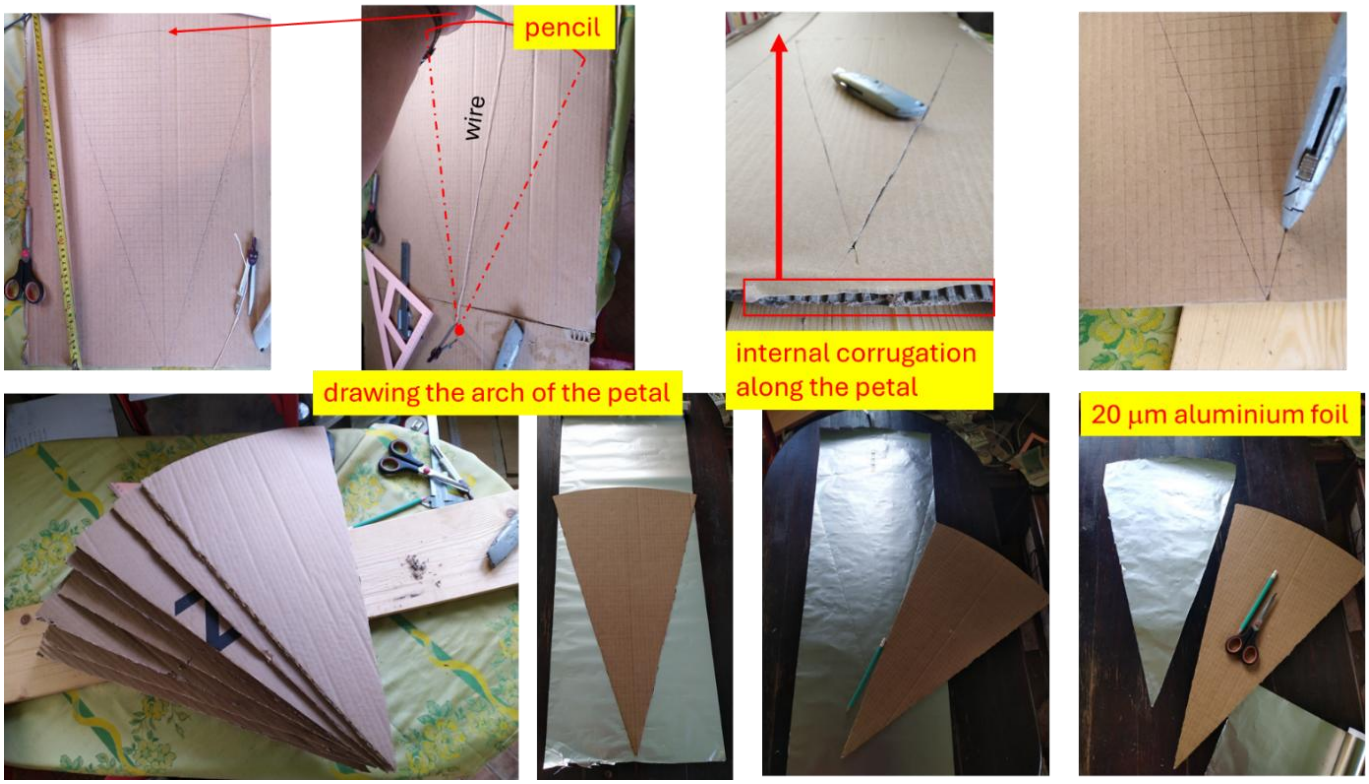


Fig. 6. Cutting of the cardboard petals from which the reflector was assembled.

The stages of gluing the paraboloid are shown in Fig. 7. Adhesive tape was used to join the petals, making the shell self-supporting. However, as noted earlier, this method does not provide perfect tangential alignment of the petals along the seams (see Fig. 5), as the tape is not rigid enough. To protect the surface from moisture, the glued surface was coated with acrylic varnish on both sides. After drying, for added strength, the inner surface was additionally lined with newspaper, while the outer surface was covered with fiberglass wallpaper cut to the shape of the petals. In the final stage, a plastic tubular frame was attached to the surface to enhance structural rigidity and allow for suspension. To reflect electromagnetic waves, the inner surface of the paraboloid was lined with segments of 20-micron-thick aluminum foil. PVA glue was used for all adhesive work.

Fig. 7. Stages of gluing the paraboloid from petals, additional surface treatment, and lining the inner surface with aluminium foil.

The assembly process of the two-tier tubular frame is shown in Fig. 8. We used short sections of plastic tubing with an outer diameter of about 10 mm. To extend the tubes, they were connected using internal couplings made from the same tubes with a longitudinal cut. The joints were wrapped with adhesive tape to strengthen the connection. A round tubular hoop with a diameter of 1 meter was stitched to the edge of the paraboloid using thin insulated wire. The radial tubes of the first and second tiers were attached to the outer hoop using end clamps (half of the tube), and at the point of intersection, they were fastened with a through bolt and washers passing through the pole of the paraboloid. Additionally, the tubes of the first tier were

secured to the surface at several points with wire to reinforce the structure. After the assembly was completed, the outer surface of the reflector was painted with white oil-based paint.

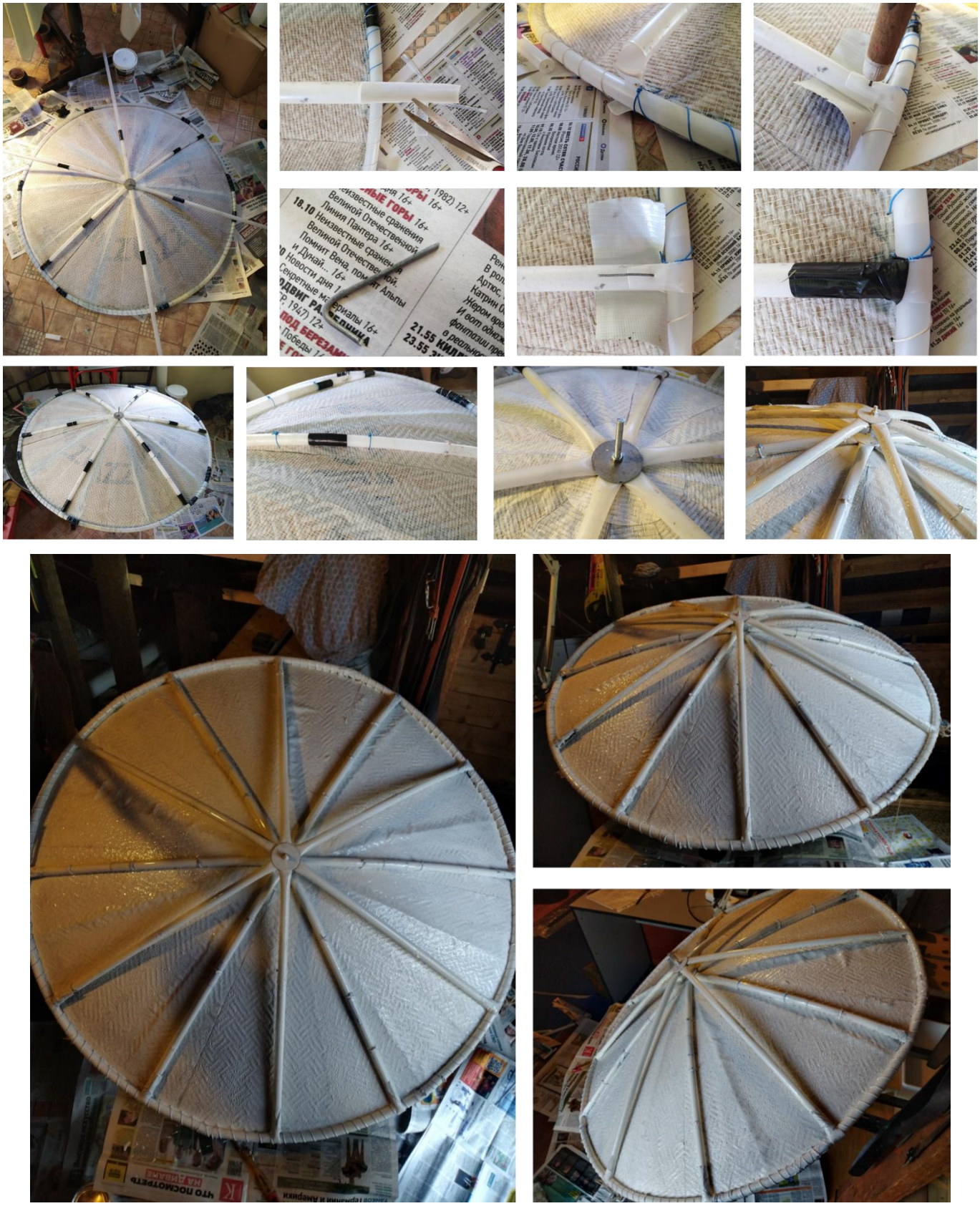


Fig. 8. Tubular frame for reinforcing the reflector and for its suspension.

The method we tested for manufacturing the reflector should not be considered as a template for future models. Rather, it helped identify issues and suggest solutions. First and foremost, experiments with materials for the petals need to be conducted. These could include thin plywood, veneer, acrylic, polycarbonate, other types of plastics, or even thin steel sheets. To ensure maximum accuracy and reproducibility, the cutting of the petals and other frame components should be performed using laser cutting, which is now accessible even through small companies for any order size. Particularly appealing is the idea of creating a single high-precision mold for subsequent production of composite impressions. The technology for producing composite shells using vacuum infusion involves layering reinforcing materials such as fiberglass or carbon fibre onto the mold, followed by impregnating them with a polymer, most often epoxy resin. The entire structure is then placed into a vacuum bag, from which air is removed, ensuring the even distribution of resin and preventing the formation of air bubbles. The vacuum aids in the deep infusion of resin into the reinforcing material, improving adhesion and the mechanical properties of the composite. The final stage is thermal processing, which completes the polymerization process. This method allows for the creation of lightweight and durable structures with high precision and uniformity.

From the above, it follows that a precise mold will be required in any case. One of the key challenges is achieving tangential alignment of the petals (see Fig. 5), which must be implemented using rigid frame ribs that pull the petal seams into a common tangent plane at every point along the parabola. Fig. 9 proposes the concept of a plywood frame capable of providing high accuracy in reproducing the paraboloid surface. All components, including the frame made from thick sheets (10– 15 mm) and thin petals (2– 3 mm, plywood or veneer), should be laser-cut. Fast-setting glue in combination with small nails can be used for securing the petals. During assembly, the petals should be pressed towards the convex surface so that the edges are aligned exactly on the centre of the ribs and along the perimeter of the outer ring. This will ensure their tangential alignment. After fabrication, the mold should be sanded along the seams and coated with waterproof varnish, allowing it to be reused multiple times for producing composite shells via the vacuum infusion method. The finished composite shell will likely need to be additionally reinforced with a lightweight frame, similar to the one we used, to prevent deformation during installation. These ideas can be further developed by continuing to experiment with materials, design, and manufacturing technologies.

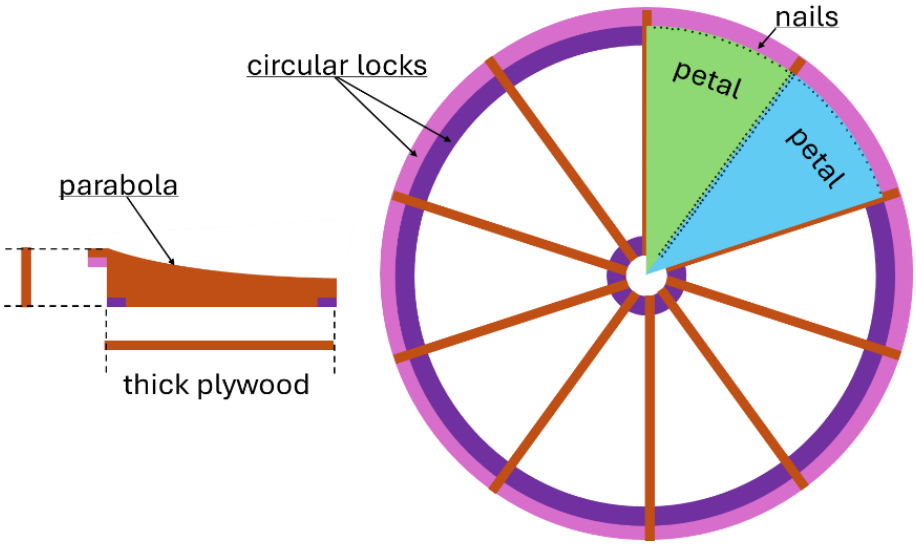


Fig. 9. Concept of a plywood frame providing high accuracy in reproducing the paraboloid surface. All components are laser-cut.

The structure shown in Fig. 9, if intended to produce composite impressions, should be considered a “negative mold”, where the impression is placed inside the concavity, with the frame positioned outside. In the case of a “positive mold”, the impression would be on the outside of the concavity, and the frame would be inside. The principle of cutting the petals remains unchanged, but pressing their edges to the ribs of the frame of a positive mold may require more meticulous work.

Positive molds may be more convenient for composite work. We would like to introduce you to a method of creating a stationary positive mold without using petal cutting, which we found on a Vietnamese craftsman's YouTube channel (although it is likely no longer an individual but a whole team). First, the outer profile of the product (made of plywood or plastic) is cut and mounted on a vertical stand with the ability to rotate, as shown in Fig. 10. A layer of raw sand is poured around this stand and levelled with the rotating profile to give it a preliminary shape. Then, liquid cement is poured over the already shaped surface and is also levelled by rotating the profile. After the cement hardens, a solid shell with the desired shape is formed. Fiberglass can then be laid on this shell and impregnated with epoxy resin. This method is worth trying in your practice. However, despite its apparent simplicity, significant skill is required to form such a shaped shell. Moreover, there are obvious drawbacks: the inability to transport the mold, difficulties with storage, and the inability to place the mold in a thermal chamber for the vulcanization of epoxy resin, which limits the ability to improve the process.

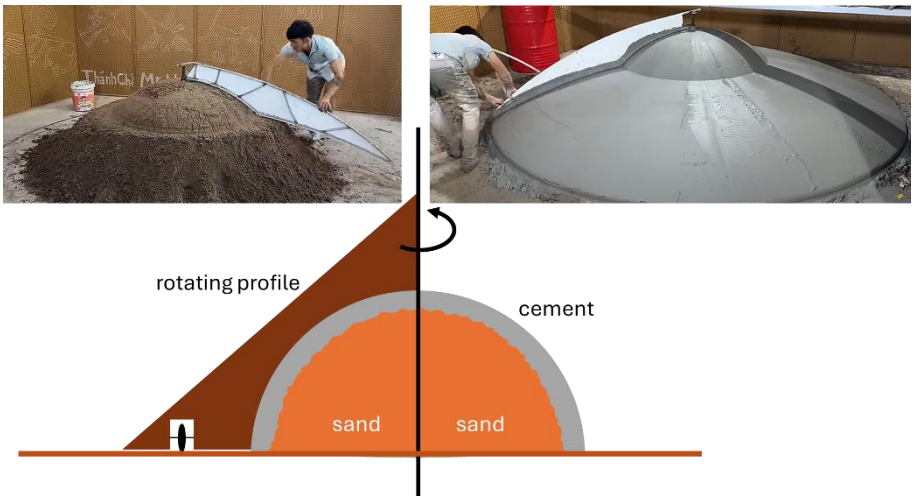


Fig. 10. Creation of a stationary positive mold from sand and cement for composite impressions.

Composite technologies have become so accessible today that they can be applied even in a small workshop. Methods such as vacuum infusion or vacuum forming do not require complex equipment and allow the creation of strong and lightweight structures. With the availability of materials like fiberglass, carbon fiber, and epoxy resins, along with the ability to use laser cutting for precise mold fabrication, manufacturing composite parts has become possible for almost any enthusiast. This opens up wide possibilities for individual projects and experimentation with new technologies.

Installation of the reflector and its orientation toward the communication mast

The reflector we fabricated turned out to be so lightweight that it could be suspended with thin ropes in the attic, fixing its direction toward the base station, as shown in Fig. 11. Therefore, we decided not to use rigid brackets at this stage of the project.

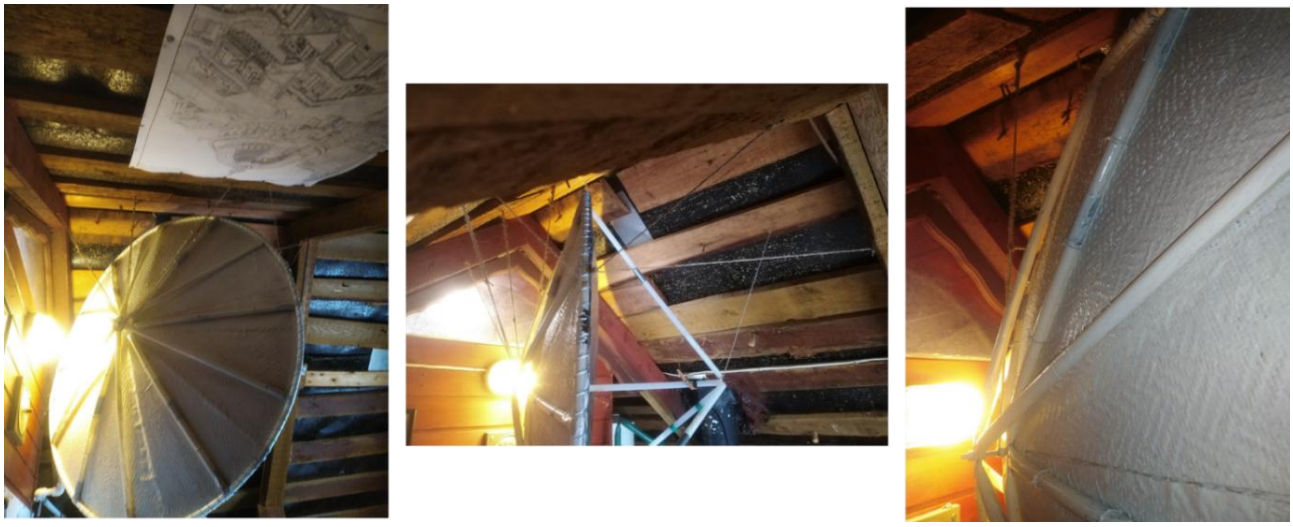
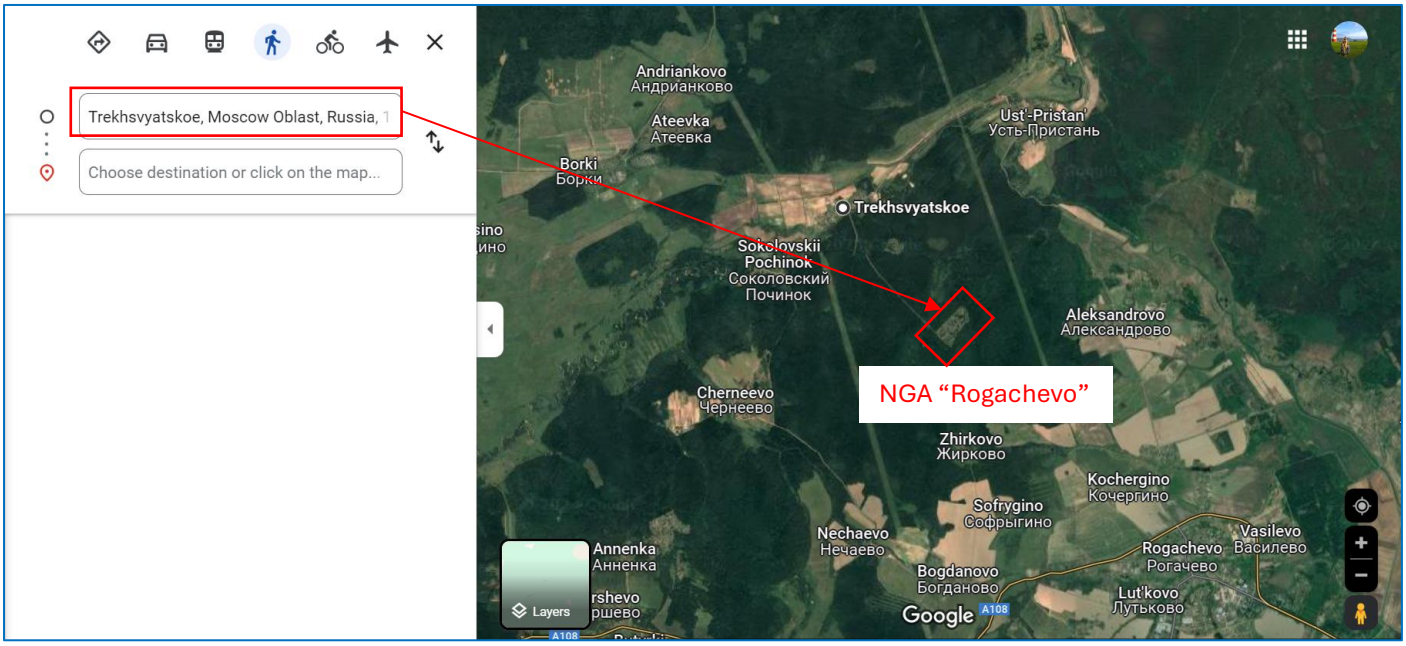


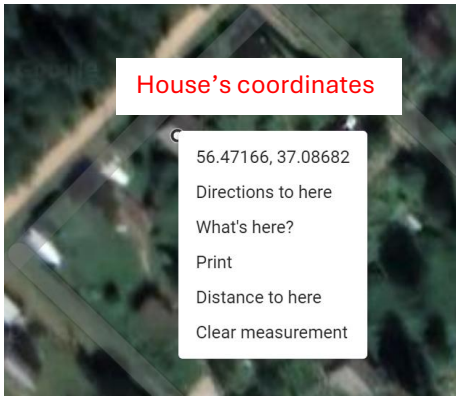
Fig. 11. Suspension of the reflector in the attic of the house.

The orientation of the reflector is carried out in several stages, as explained below. First, using Google Maps, locate the gardening association, which should be visible from satellite view.

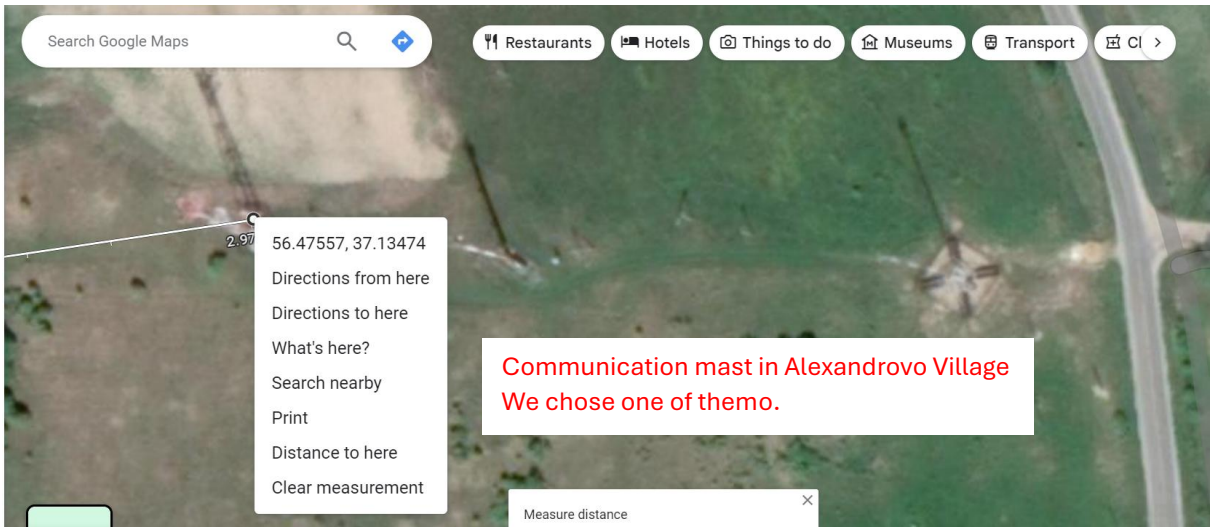


Zoom in, and you will be able to see your plot and even your house. Using the right mouse button, open the menu shown below and select “measure distance”. A reference point will appear, which can be moved with the mouse to your building where the antenna is installed.

Move the mouse cursor over this point and right-click again. In the opened menu, you will see the geodetic coordinates of your house.



Without moving the first reference point, zoom out (using the mouse wheel), navigate the map to the area where the communication mast is located (by dragging the map with the mouse), zoom in again, and find the mas. Then, using the right mouse button, open the menu and select “distance to here”. A second reference point will appear, which you need to move to the communication mast. Next, by pointing the mouse cursor at this point and right-clicking, you will obtain the geodetic coordinates of the base station.



Zoom out (using the mouse wheel) to see the full direction and distance from your house to the tower.

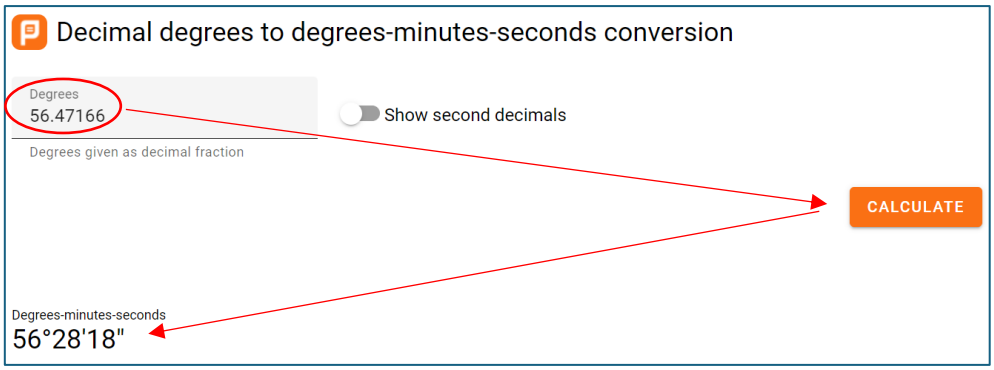
So, we have determined:

The distance from house 62 to the communication mast – 2 km 970 m.

Geodetic coordinates of house 62 – (56.47166, 37.08682)

Geodetic coordinates of the mast in Alexandrovo Village – (56.47557, 37.13474)

Now, it is necessary to convert the fractional (decimal) values into degrees, minutes, and seconds, which can be done on the website, as shown below.

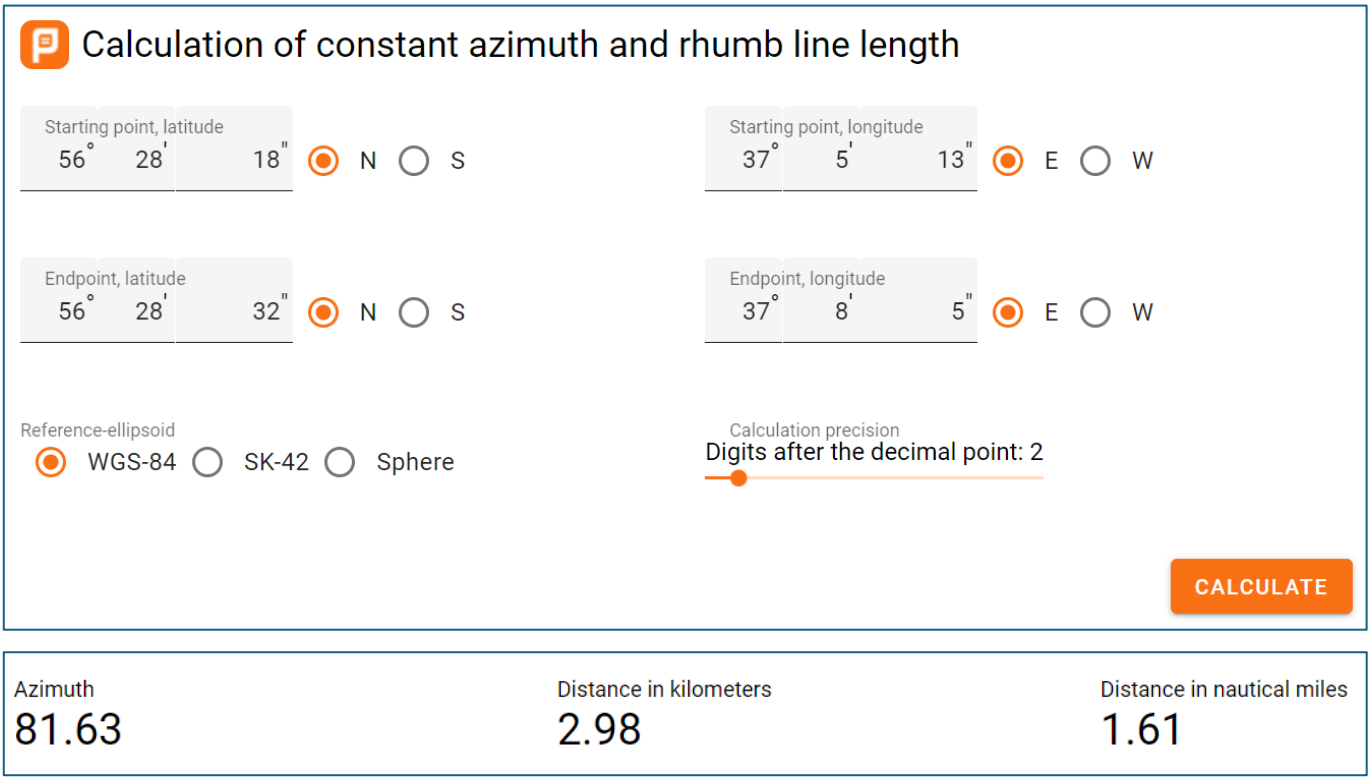


As a result, we will obtain:

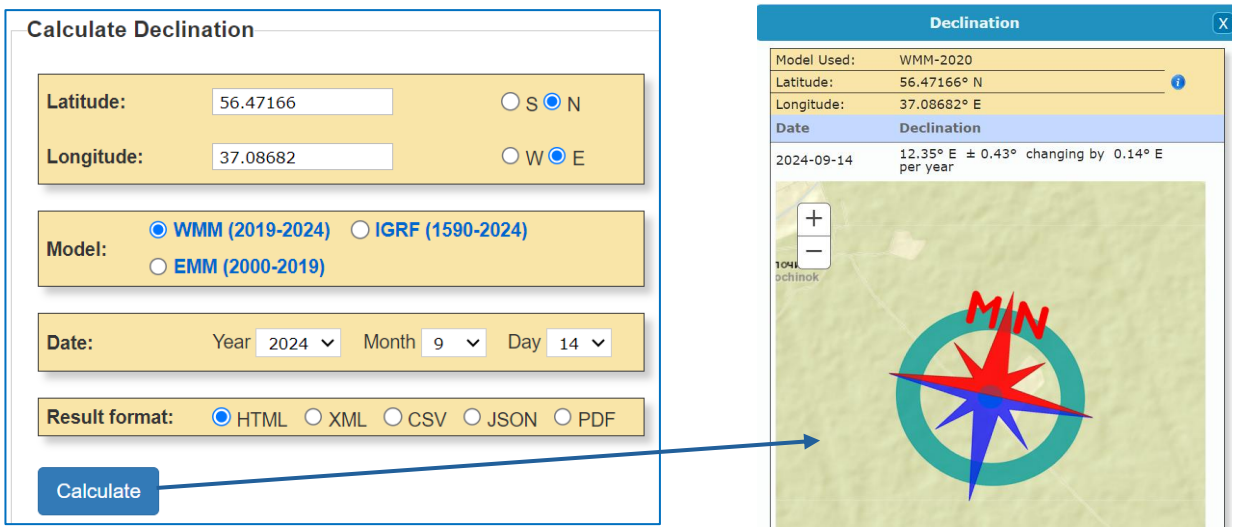
Geodetic coordinates of house 62 – 56°28′18" N.Lat. and 37°5′13" E.Lon. (North Latitude and East Longitude).

Geodetic coordinates of the mast in Alexandrovo Village – 56°28′32" N.Lat. and 37°8′5" E.Lon.

Using these coordinates and the website, we can calculate the azimuth to the communication mast, as shown below.

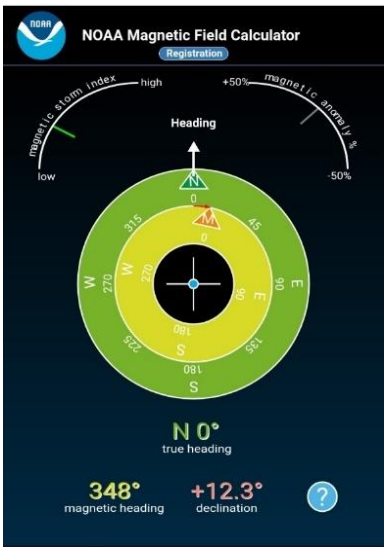


We have determined that the geographic azimuth from house 62 to the mast in Alexandrovo Village is 81.63^0 to the East. However, if we want to measure this azimuth using a compass, we must account for magnetic declination, as Magnetic North does not coincide with Geographic North. Magnetic declination can be calculated on the National Centres for Environmental Information website by entering your coordinates in decimal form, as shown below.



For our NGA “Rogachevo”, the magnetic declination is approximately 12.35^0 to the East, and this value “drifts” by 0.14^0 East each year. Therefore, when reading this report, please verify the current value on the website, although such high precision is not necessary for our purposes. Thus, the magnetic azimuth from house 62 to the mast in Alexandrovo Village is $81.63^0 - 12.35^0 = 69.28^0$. This is the azimuth that will be used for aligning the reflector. It is important to note that this direction will vary from house to house, as the area of NGA is not insignificant compared to the distance to the mast.

On a smartphone, magnetic declination can be determined automatically, as it will detect your current geolocation. Simply search for the NOAA Magnetic Field Calculator in your browser. Below is a screenshot from a smartphone, taken at the corner of our NGA, where internet was still available. From it, we can see that Magnetic North M is offset relative to True North N by $+12.3^0$, where a positive sign indicates East.



Directional antennas, especially highly directional ones like a parabolic reflector, require precise alignment with the base station, rather than focusing on a local interference peak, which may change over time. Therefore, such antennas should not be rotated around their axis, but rather their direction should be adjusted within small deviations from the calculated azimuth to the base station. The higher the antenna is positioned, the more accurately it can be aligned. A smartphone can be used to measure the signal strength from the modem, but its readings may have a delay, making this method less precise. Therefore, adjustments to the direction should be made very slowly. Ideally, an electromechanical drive would be used to control the antenna's rotation, but this is not accessible to everyone. In most cases, manual adjustment must suffice.

Our manual method for aligning the reflector with the communication mast using the magnetic azimuth is shown in Fig. 12. The reflector was equipped with front frames made of plastic tubes that clearly mark its focus, where the modem is attached. A compass with azimuth markers showing the direction of 69^0 East was mounted on the central horizontal tube. By rotating the suspended reflector, we aligned the magnetic needle with North. This direction was then secured with additional ropes.

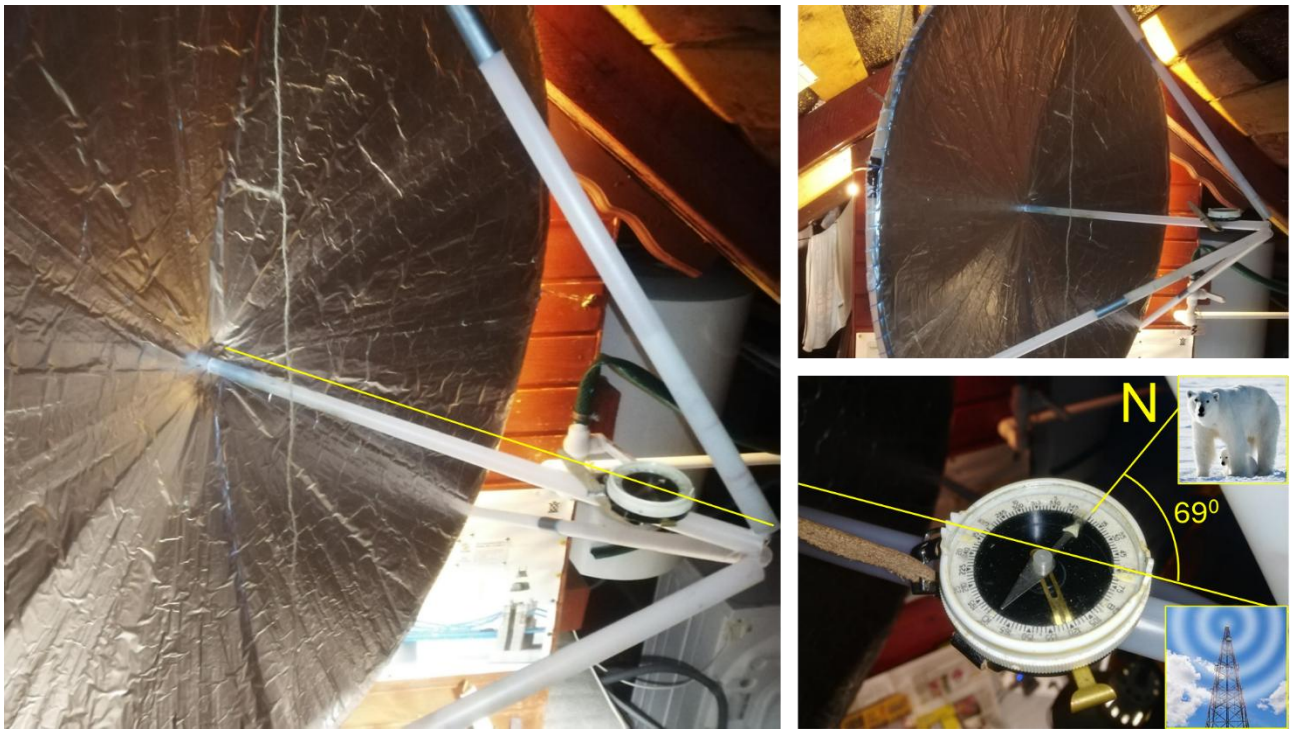


Fig. 12. Manual alignment of the reflector with the communication mast using the magnetic azimuth.

Equipping the reflector with a 4G modem and feed

A parabolic reflector, used in telecommunications applications, performs both signal reception and transmission functions. The reflective surface itself efficiently works with electromagnetic waves of any polarization. However, the reflector is highly sensitive to its orientation relative to the base station, as it has a narrow radiation pattern. The reflector can be equipped with various types of feeds available on the market, providing ample opportunities for experimentation. With sufficient knowledge, one could even attempt to design a custom feed horn.

In transmission mode, the feed receives a signal from the modem via a cable and forms a radial electromagnetic wavefront with a specific polarization. This wavefront, when incident on the parabolic surface, is reflected, creating a directional beam parallel to the reflector's axis, which is directed towards the base station. In reception mode, the feed captures the electromagnetic radiation focused by the reflector and converts it into a signal, which is then transmitted to the modem via the cable.

Due to the focusing of waves from the large aperture of the reflector onto the small aperture of the feed at its focus, signal power is amplified. The power gain of the reflector (in dB) is calculated by the following equation:

$$G = 10\ln\left(k\left(\frac{2\pi R}{\lambda}\right)^2\right)\ \text{dB} \qquad (14)$$

where R is the reflector radius, λ is the wavelength, and k is the efficiency coefficient of the reflector. This equation has already been utilized in the program:

```
39. # Antenna gain range [Gain_min, Gain_max] dB for the given reflector efficiency k
40. Gain_min = 10.0 * np.log10(k * (2.0 * pi * radius / lambda_max)**2)
41. Gain_max = 10.0 * np.log10(k * (2.0 * pi * radius / lambda_min)**2)
```

Let us note that perfect focusing to a single point is impossible in the context of wave optics, where the wavelength is not infinitesimally small, unlike geometric optics that assumes ideal light rays. In wave optics, the focus is represented as a volume with a transverse dimension approximately equal to the wavelength λ. The smaller the wavelength (and correspondingly higher frequency), the smaller the focal volume, which results in an increased gain factor. For the 4G standard, a wide range of frequencies is allocated, from 900 MHz to 2.6 GHz, which corresponds to wavelengths ranging from approximately 0.33 m to 0.12 m. Accordingly, the signal gain can vary widely depending on the carrier frequency.

The coefficient k, as indicated in eq. (14), depends, among other factors, on the accuracy of the paraboloid's shape reproduction and the smoothness of its surface after foiling. In our cardboard reflector, these parameters were of mediocre quality. Before, we have already outlined potential solutions to these issues. The design of the feed horn, which is a passive antenna, will also influence the value of k:

- The feed must be designed so that its shape and size match the geometry of the reflector. Mismatch can lead to improper radiation distribution and, consequently, reduced efficiency. For example, if the feed is too large or has an incorrect shape, it may create areas with uneven energy distribution, which will negatively impact the reflection efficiency.
- Precise positioning of the feed relative to the reflector is critical. If the feed is not positioned at the focal point of the paraboloid, the radiation will not be efficiently concentrated, resulting in energy loss and a reduction in the overall efficiency factor.
- The feed must be broadband to cover all possible frequencies used for communication.

Modern wireless communication systems require high data transfer rates, reliability, and efficient use of the frequency spectrum. To achieve these goals, various signal transmission schemes are used, including SISO (Single Input Single Output) and MIMO (Multiple Input Multiple Output). These transmission schemes differ significantly in architecture, performance, and areas of application.

SISO is a traditional signal transmission scheme, where the transmission is carried out through a single antenna on the transmitting side and a single antenna on the receiving side. It is the simplest and most basic method of data transmission in wireless systems. In a SISO system, the transmitter modulates the data onto a single carrier frequency and transmits it through one antenna. The receiving antenna then captures the signal, which is demodulated to extract the transmitted information.

Advantages of SISO:

- SISO systems require fewer hardware components (one antenna and one RF chain), which reduces both the cost and complexity of the system.
- A single transmitting antenna requires less power compared to MIMO systems.
- Signal processing in SISO systems is simpler, which reduces the computational resource requirements.

Disadvantages of SISO:

- Since data is transmitted through a single antenna, the throughput is limited by the transmission rate of that single antenna.
- In the case of multipath propagation, significant losses can occur due to signal interference, which reduces the reliability of transmission.
- Limited capability to increase data transmission rates compared to more advanced schemes like MIMO.

MIMO is a transmission scheme in which multiple antennas are used both on the transmitting and receiving sides. This scheme allows multiple data streams to be transmitted simultaneously, thereby increasing data rates and improving signal quality. The main concept of MIMO is to utilize spatial multiplexing and/or signal processing to enhance spectral efficiency and resistance to fading. In MIMO systems, multiple data streams can be transmitted simultaneously through different antennas, which increases the throughput without expanding the frequency band.

Advantages of MIMO:

- The ability to transmit multiple data streams simultaneously allows for a significant increase in transmission speed.
- Using multiple antennas improves resistance to fading and interference, as each data stream travels through an independent channel.
- MIMO allows for more efficient use of the available frequency spectrum.

Disadvantages of MIMO:

- MIMO systems require additional hardware (multiple antennas and RF chains), which increases the cost and complexity of development and deployment.
- The presence of multiple transmitting and receiving antennas increases energy consumption.
- More powerful hardware and more complex signal processing algorithms are required to handle signals and mitigate interference.

Modems and communication systems, such as 4G LTE and 5G, use adaptive MIMO, which means dynamically switching between transmission and reception modes based on current communication conditions. MIMO transmission modes include:

- *Spatial Multiplexing*: Transmission of multiple independent data streams through different antennas, which increases the overall data rate. The signal from each antenna does not travel in a straight line. In a real environment, the signal reflects off numerous objects such as walls, buildings, vehicles, and trees. This causes the same signal to travel along multiple distinct paths before reaching the receiving antenna. These reflected signals can have different phases, amplitudes, and delays due to variations in path length. Although the transmitting antennas may be quite close to each other (e.g., within a few centimeters in a modem), the environment surrounding the antennas plays a crucial role. Each signal transmitted by a separate antenna encounters different objects and irregularities in the surrounding environment, resulting in multiple paths of multipath propagation. Thus, while the antennas may be close to each other, the paths their signals take can vary significantly. When these different paths arrive at the receiving antennas, they exhibit different time delays, phases, and signal levels. The receiving system (e.g., in a smartphone or base station) employs complex signal processing algorithms to differentiate these data streams and reconstruct the original transmitted information. Spatial

multiplexing is effective in conditions where there is a significant degree of multipath propagation, allowing each data stream to follow distinct trajectories. In an urban environment, with many reflections from buildings and other objects, this condition is generally met, even if the antennas are only a few centimetres apart.

- *Diversity Gain*: The use of multiple antennas to transmit the same signal with different characteristics in order to reduce the probability of signal loss due to fading. Unlike spatial multiplexing, the goal of diversity gain is not to increase data transmission rates but to improve the system's resilience to signal loss. In MIMO systems with diversity transmission, each transmitting antenna sends the same data stream but with certain variations (e.g., different delays or phases). This is done to ensure that the signals travel through different spatial paths, increasing the likelihood that at least one of them will reach the receiver with good quality.
- *Beamforming*: Directed transmission of a signal using phase shifts at the transmitting antennas to concentrate the signal power in a specific direction.

The choice between modes depends on the quality of the communication channel, and the modem makes this decision based on current radio channel conditions using specialized adaptation algorithms:

1. Channel State Information, CSI
   - RSSI — Received Signal Strength Indicator
   - BER — Bit Error Rate
   - SNR — Signal-to-Noise Ratio
   - Multipath Distortion (or Fading Depth)
   - Interference Level (impact of other signals)

   These parameters are transmitted to the base station or receiving device through dedicated control channels, and based on this information, the modem decides which transmission scenario to use.
2. Choice Between Spatial Multiplexing and Diversity Transmission
   - Spatial Multiplexing is used when the channel quality is good. This means that there are independent propagation paths (different multipath trajectories) between the transmitting and receiving antennas, and each data stream can be transmitted without significant distortion. In this case, multiple data streams can be transmitted simultaneously, thereby increasing the overall capacity.
   - Diversity Transmission is used when the channel quality is moderate or poor. If the signal experiences severe fading or interference, the modem switches to diversity transmission to improve communication reliability. In this case, instead of transmitting multiple data streams simultaneously, the modem transmits the same stream through multiple antennas. This helps compensate for potential losses, as the receiving side can select the best signal from several received copies.

Miniature and inexpensive stick modems without external coaxial connectors for antennas are almost certainly of the SISO type. As part of testing, we purchased one of these modems, see Fig. 13, on the Wildberries marketplace. The USB connector is used exclusively to power the modem. It can also be connected to a charger, where the current must be at least 1.5 A, otherwise the modem will not function. Once the modem is powered on, configuration is done through a web interface in a browser, accessible via a Wi-Fi connection. The connection address, which must be entered in the browser's address bar, is specified by the manufacturer in the manual. In our case, it was 192.168.100.1. On the opened webpage, you need to enter a login and password (which can be changed), after which the settings can be configured.

Fig. 13. 4G stick modem without external antenna connectors.

The use of such a modem should be rejected for several reasons: instability, poor build quality, and the inability to connect an external antenna (which will be required in the next stages of the project). First, it was found that the modem has extremely unstable Wi-Fi, which disconnects at regular intervals. Then, after a few connections, the USB connector became loose. To determine the cause, we opened the modem's cover and found a so-called “cold solder joint” on the printed circuit board — a situation where the soldering process failed to create adhesion between the contact and the solder, as shown in Fig. 14. We had to resolder the entire USB connector.

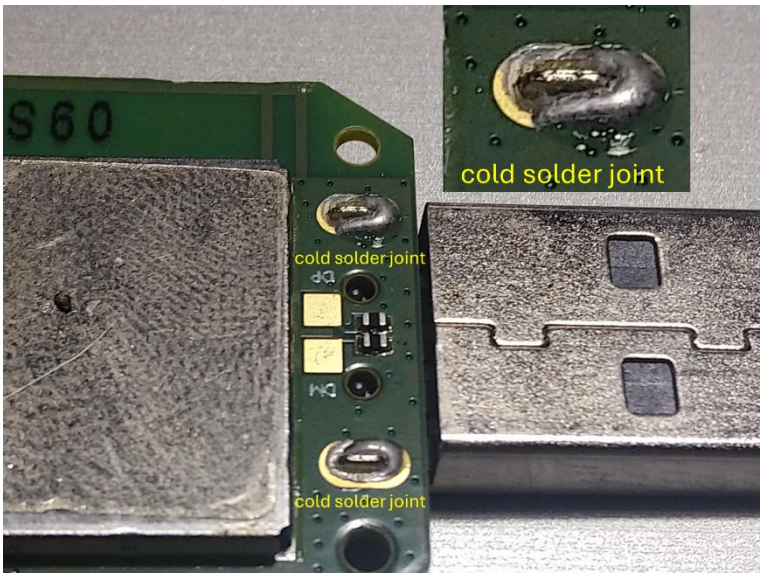


Fig. 14. Example of a "cold solder joint" on the printed circuit board of the stick modem. It is visible that the solder does not envelop the contacts.

Upon opening the modem's casing, petal contacts intended for a single antenna were discovered (see Fig. 15), located on the plastic cap. This confirms our assumption that the modem is of the SISO type. The operating principle of the antenna could not be determined. Measurements with a multimeter revealed that the "GND" and "ANT" contacts are short-circuited, which was surprising. On the inside of the cap, there is a square plate (see Fig. 15), which possibly serves as a patch antenna, although both petal contacts touch its surface. The purpose of the black adhesive strip on the cap also remains unclear. Thus, the question regarding the construction of the antenna input remains open.

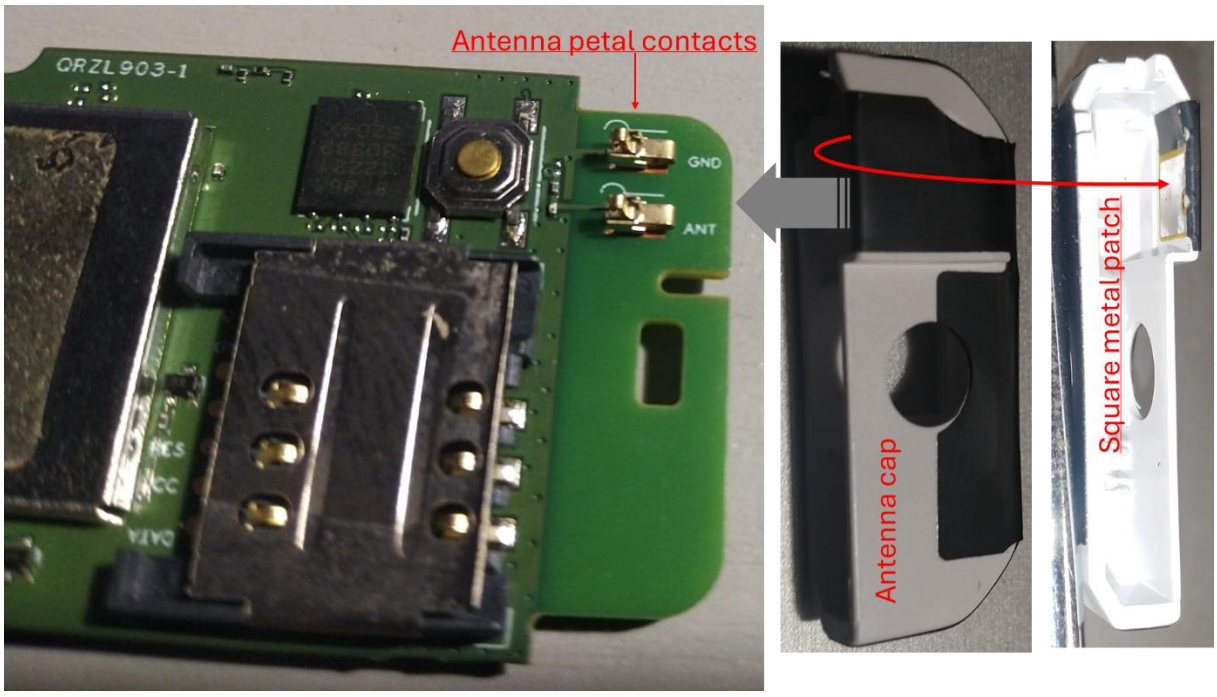


Fig. 15. Internal SISO antenna of the modem.

Our second attempt involved a 4G stick modem of the SISO type from the telecom operator MegaFon (see Fig. 16), using its SIM card. We first ensured that MegaFon provides 4G coverage in the area of our garden community. Additionally, the operator's website indicated the presence of a base station in the village of Alexandrovo, which we used to aim the reflector. This modem represents a completely different level of quality, including a user-friendly browser interface, although its cost is four times higher compared to the previous device. While all of these electronics are produced in Shenzhen (China), the brand ordering the product plays a key role, including quality control. The first modem was unbranded, with cost-cutting measures on everything, including components, assembly, and software. The MegaFon modem can operate autonomously, but it also has a coaxial input for an external SISO antenna.

Fig. 16. 4G stick modem of SISO type from the telecom operator MegaFon.

At the initial stage of the project, we undertook a rather bold, though technically unjustified attempt — placing the MegaFon stick modem at the focal point of the reflector, as shown in Fig. 17. The modem was powered by a charger via a USB extension cable. Understanding the principles of how a feed (directional antenna) and a reflector operate, this setup is subject to serious criticism. First, we have no information about the radiation pattern of the modem, making it difficult to choose its optimal orientation at the reflector's focal point. Second, this radiation pattern is clearly not designed to work with a reflector. Moreover, the polarization of the incoming wave, focused on the modem, may not match the orientation of the modem's internal antenna, leading to additional signal losses and reduced reception efficiency. Due to these factors, the overall system efficiency cannot be high (low value of coefficient k in eq. (14)). Nevertheless, we managed to achieve a signal gain of around 10 dB, although with a proper feed — even with moderate reflector efficiency ($k = 0.6$) — a gain of 17 to 26 dB could have been expected. The result was also undoubtedly affected by the

insufficient accuracy of the paraboloid surface reproduction. All of this will be considered in the next stages of the project.

Let us reiterate that the decision to install the modem at the focal point of the reflector without a feed was not due to incorrect assumptions, but rather aimed at exploring and justifying the configuration of the future system. This included developing the technology for reflector fabrication, selecting the modem, and determining the appropriate feed. Readers of our report who wish to implement the proposed solutions will not need to repeat this initial stage.

Fig. 17. MegaFon 4G stick modem installed at the reflector's focal point.

The next step in improving the system with the reflector should be connecting an external feed of either SISO or MIMO type, depending on the modem used. Both feeds are broadband directional antennas. The SISO feed uses a single antenna, whereas the MIMO feed includes two antennas (and correspondingly, two output cables) with cross-linear polarizations to minimize crosstalk (mutual interference) between them. For a compact stick modem, it could be placed next to the feed and connected with short cables to minimize signal transmission losses. This configuration is shown in Fig. 18 for a MIMO modem (already a commercially available solution). In the case of using a SISO modem (with a single cable), a MIMO feed can still be used,

but the unused output on the feed (of either polarization) should be terminated with a 50 Ω coaxial load (SMA or N-type connector) with an operating range up to 6 GHz, which is easy to find and purchase.

Nevertheless, the configuration presented in Fig. 18 does not seem entirely optimal to us. The reason is that at the focal point of the reflector, which amplifies the signals, there are heterogeneous sources of radiation, such as the feed and the modem itself, which might negatively impact the performance of the MIMO communication channels during data transmission due to interference, although the exact impact is unknown. It is possible that a configuration with a SISO feed would be less sensitive to the system’s setup. Given these concerns, it seems advisable to place the modem outside the irradiation zone of the reflector, for example, behind it. In this case, the coaxial cables would remain relatively short, no longer than one meter. Examples of 4G MIMO-type modems that can be used for such a configuration include the ZTE MF283 (supports up to 32 users) or the Huawei B818-263 (supports up to 64 users). Both modems are equipped with two coaxial connectors for connecting external antennas. The Huawei modem offers greater functionality, including the ability to use a VPN connection, LAN and telephone ports, and it also retains the SIM card PIN code after power is disconnected. This feature is particularly important for use at a cottage, where power outages are possible. Otherwise, as in the case of the MegaFon modem, it would be necessary to re-enter the device settings in the browser and input the PIN code (0000 or 1111 if it has not been changed to a unique one) every time power is restored.

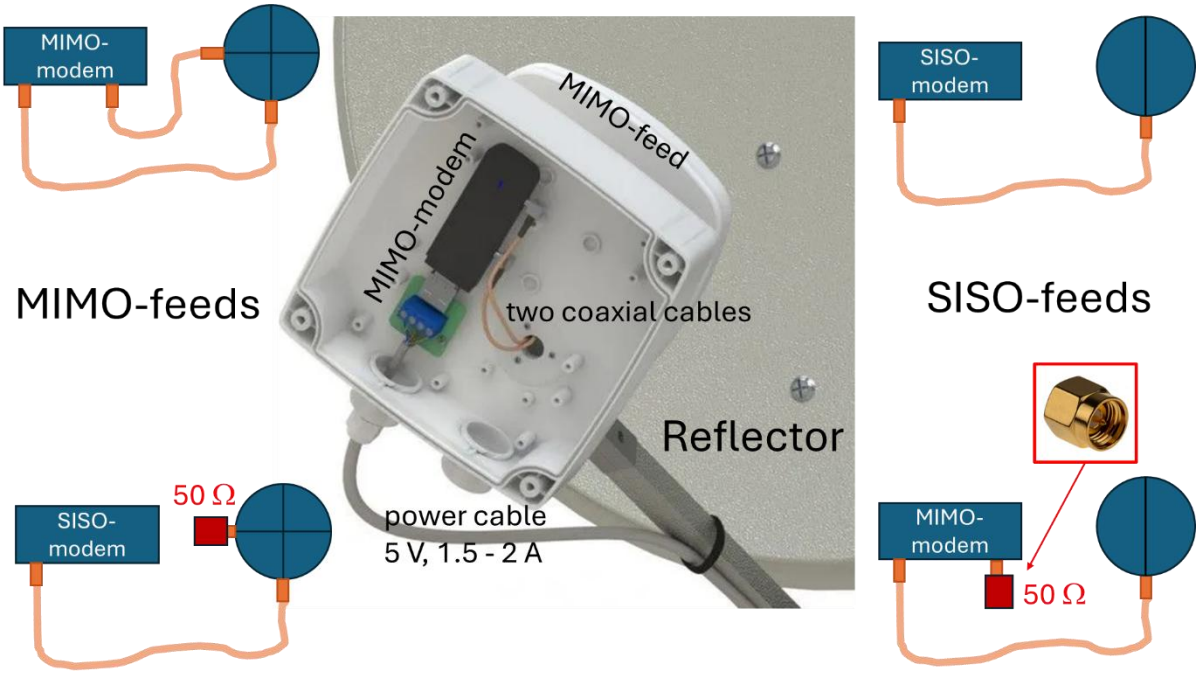


Fig. 18. Installation diagram of a stick modem with a feed, as well as various connection schemes between the modem and feed depending on their types.

When choosing a feed, which is essentially a directional antenna, it is important to consider the operating frequency range to ensure that it fully covers the frequencies used by your provider for 4G communication. The broadband MIMO-type feed we found on AliExpress has a range from 698 to 6000 MHz, allowing it to be used with any modem. To connect cables to the modem, coaxial adapters may be required, which can also be easily purchased on AliExpress. This feed comes with a metal mounting rod that sets the recommended focal distance for a truncated mesh reflector with dimensions of 90 × 60 cm², with which it is typically used, as shown in Fig. 19.

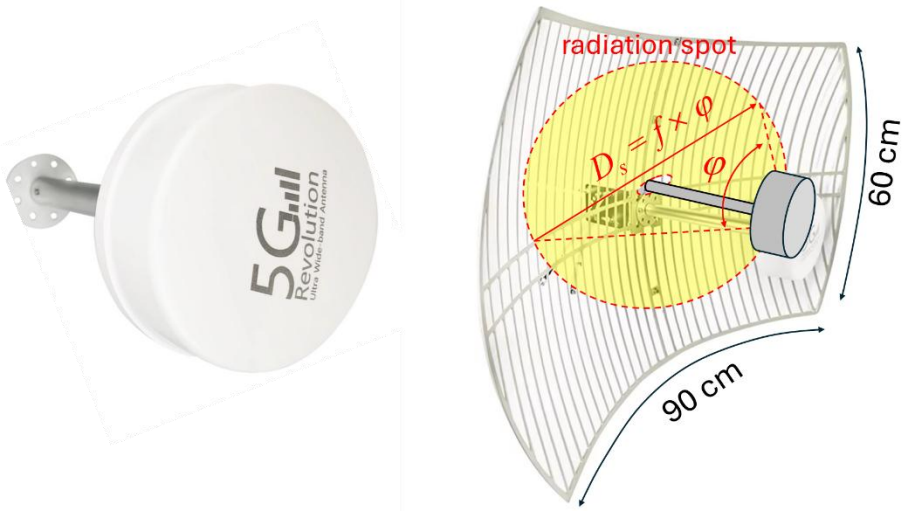


Fig. 19. Example of a broadband MIMO feed mounted on a mesh reflector.

The first question that arises concerning this feed is — in which direction does it radiate? The manufacturer has provided no information (Chinese, unbranded), and we also do not know the details of its design. Typically, feeds direct the signal towards the reflector for transmission and receive the reflected and focused signal back. However, the metal mounting rod originating from the centre of the feed's casing raises some doubts, as it may create distortions for the radiation. Nevertheless, we will assume a classical setup where the feed radiates towards the reflector. The feed forms a radiation pattern that includes a main lobe along the axis, as well as side and back lobes that do not contribute to the signal transmission. The gain of the feed is determined by the ratio of the energy radiated in the main direction to the energy radiated in other directions. On the reflector surface, the main lobe with an opening angle φ (in radians) forms a "spot" with a diameter $D_s = f \times \varphi$, where f is the focal length. The size of the spot depends on the frequency: the lower the frequency, the larger the opening angle. For the reflector to work effectively, the spot must be within its boundaries at any frequency to avoid a situation where the feed radiates beyond the reflector. This requirement determines the minimum size of the reflector. Therefore, when fabricating the reflector, it is very important to know the maximum opening angle of the main lobe at the lowest frequency in the feed's operating range. In the feed description, the manufacturer provided horizontal and vertical angles for the main lobe — 9^0 and 8^0, respectively. This is a very narrow beam (slightly asymmetrical). Since, as noted earlier, these angles depend

on frequency, it is most likely that they refer to the maximum frequency of 6 GHz. For the lower frequency of 698 MHz, the angles will be significantly wider. Without reliable data, it is nevertheless possible to estimate the maximum possible opening angle of the main lobe (in radians) by dividing the minimum reflector size of 60 cm by the length of the rod (the focal length).

In Fig. 20, we show another interesting solution for a feed from Waveform. Chinese unbranded equivalents can be found on AliExpress at a much lower cost. However, we cannot guarantee their quality or compliance with the stated specifications. The technical specifications of the antenna and the installation guide are available on the company's website. The feed is a broadband (600 – 6500 MHz) SISO antenna (single cable) with a narrow radiation pattern directed towards the reflector.

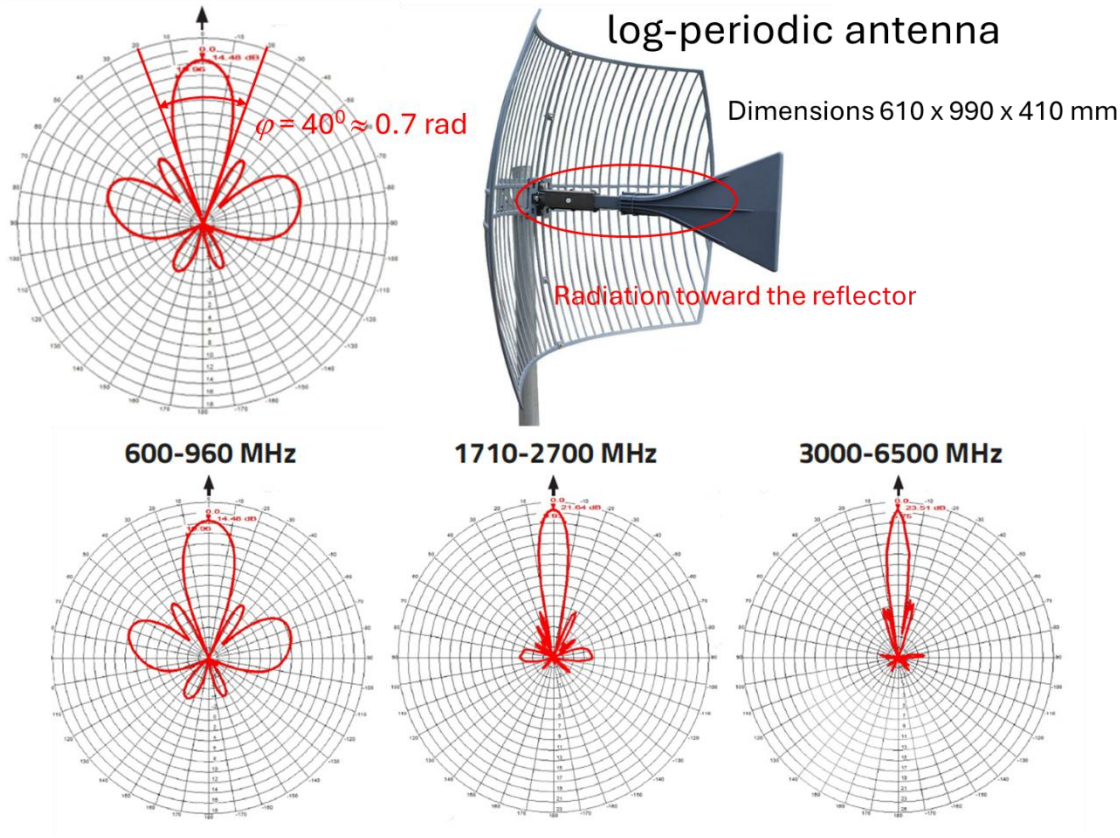


Fig. 20. Broadband SISO feed mounted on a mesh reflector. radiation patterns (in red) refer only to the feed (log-periodic antenna), not the entire system.

The type of antenna (hidden within the protective casing) was confirmed directly by the company's engineers—it is a so-called “log-periodic antenna”. It is possible that it is implemented on a printed circuit board, but we cannot say for certain. However, this type of antenna is well-documented in the literature, so it could potentially be self-fabricated. Note that this antenna is also mounted on a central metal rod, similar to the previous feed. However, this design choice did not raise objections from the designers, whom we are inclined to trust.

This antenna radiates towards the narrowing end, i.e., towards the reflector. From the radiation patterns provided by the company (see Fig. 20), it is possible to estimate the maximum size of the radiation "spot" on the reflector by taking the maximum opening angle of the main lobe, $\varphi \approx 0.7$ rad, and the antenna length of 410 mm: $D_s = 0.7 \times 410 \approx 290$ mm, which is about half the vertical size of the reflector (610 mm). This is as it should be — the reflector size should exceed the maximum "spot" size.

The rod on which the antenna is mounted has four fixed positions along its axis, see Fig. 21. As explained in the installation guide, changing the position can increase the gain by 2 dB for the selected frequency range. If this slight improvement is not critical for you or if you do not know the operating frequency range, it is recommended to set the rod in position "3". The very presence of such an option indicates that the antenna has been carefully designed and tested, which inspires confidence in its stated specifications.

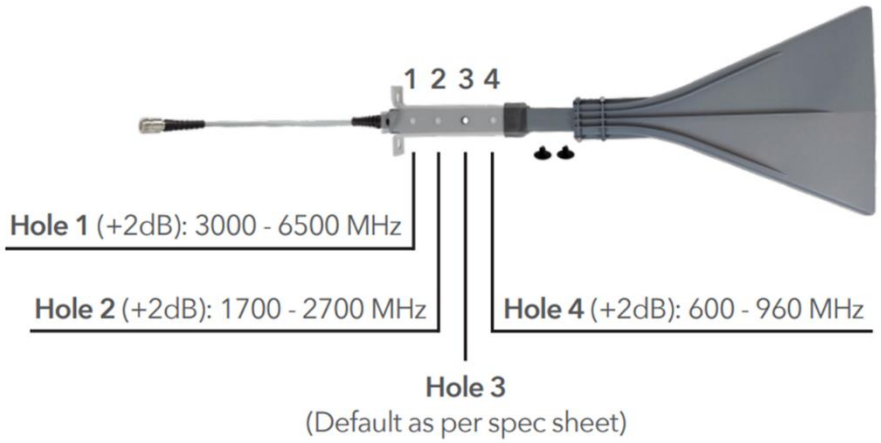


Fig. 21. Antenna gain adjustment for the selected frequency range by moving it to fixed positions on the rod.

In a MIMO feed, which has two antennas with crossed polarizations, the polarization of incoming and focused electromagnetic waves will always match one of the antennas or partially align with both. This means that regardless of the polarization of the incoming wave, it will be effectively received by at least one of the antennas. Such a configuration reduces sensitivity to the antenna's orientation around its axis and improves the overall efficiency of signal reception. In contrast, a SISO feed with a single antenna is more sensitive to its orientation around its axis. If the polarization of the incoming wave does not match the polarization of the antenna, the signal reception efficiency may decrease. Therefore, for SISO feeds, proper orientation is crucial for maximizing reception quality. The specific feed we are considering does not support independent rotation around its axis, but it can be rotated together with the reflector during alignment with the communication tower.

The installation guide also discusses the use of two crossed reflectors, see Fig. 22, for connection to a MIMO modem. Such a configuration replicates the crossed polarizations found in MIMO feeds. The setup with two reflectors and SISO feeds deserves careful study, and we consider it to be a highly promising approach for our project.

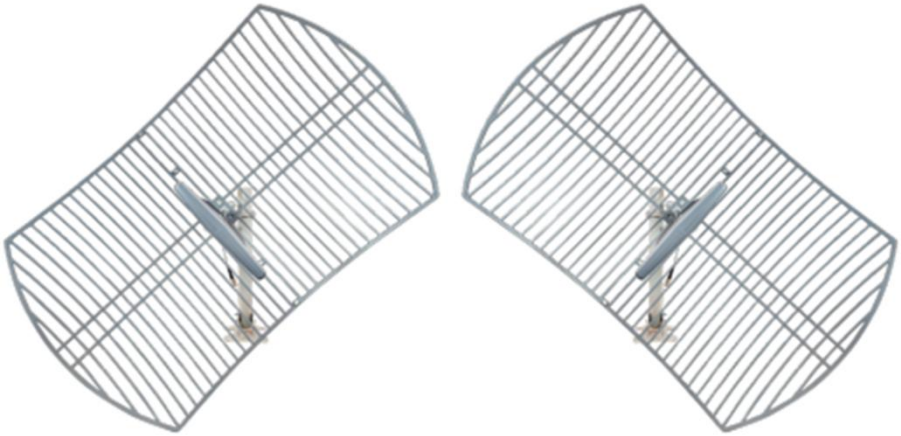


Рис. 22. Использование двух скрещенных рефлекторов для подключения к MIMO-модему.

From our brief overview of MIMO modems (see page 24), we know that there are three data transmission modes, of which only one—spatial multiplexing—provides an increased data rate. In the other modes, the modem prioritizes channel stability over data speed. To switch the modem to this mode, it is necessary to (1) ensure a high signal level and (2) provide two uncorrelated transmission channels. When using a single MIMO feed with closely spaced internal antennas, the establishment of two distinct channels may not occur, whereas with a two-reflector setup, this is more likely. Therefore, a two-reflector system with SISO feeds can indeed be more effective for implementing the spatial multiplexing mode in MIMO systems. This is due to the reduced correlation between channels and the improved conditions for transmitting independent data streams.

Regardless of the chosen feed, we will be manufacturing a custom-designed reflector of significantly larger diameter (up to 1.5 m). When increasing the reflector diameter $D_r = 2R$ (see eq. (2)), it is necessary to maintain the recommended f/D_r ratio for the given feed, where f f is the focal length. Typically, this ratio ranges from 0.5 to 0.9 (in our cardboard reflector, it is 0.5). Reputable feed manufacturers will specify the recommended f/D_r ratio. Pay attention to the operational frequency range, which should fully cover the entire 4G spectrum, and ensure that the feed has a 50 Ω impedance to match the modem (most microwave equipment is designed for 50 Ω).

We are also interested in developing our own feeds based on well-established designs published in books and scientific journals. We have access to such literature. In this report, we do not delve into antenna theory and electronics, which will be necessary for the development of advanced reflectors or an entire receiving station. These are topics for the next report. However, based on this report, one can already begin manufacturing a custom reflector and selecting suitable commercially available modems and feeds.

General principles of building a mast-based receiving station

Several factors make one consider the necessity of installing a mast-based receiving station. Everyone understands that the antenna should be placed as high as possible, which is especially important in wooded areas. In principle, the reflector, even of large diameter, can be installed in the attic, as in our case, provided the roof is non-metallic. However, in some cases, such placement is unacceptable — for instance, due to living spaces in the attic or a lack of available space. In such situations, the reflector will have to be mounted outside. Since this is not a satellite dish pointed toward the sky, it will need to be positioned high on the gable of the house, oriented toward the communication tower. But if the orientation of the house does not allow for such placement, additional challenges arise. Installing the reflector on a standard pipe mast is also not suitable due to its large wind resistance and the challenging weather conditions (think of icy rains in winter). Rusty reflectors on house facades have become a common sight, but these are typically small satellite dishes, more resistant to environmental impacts. So, what solution can be proposed in such a case?

If installing the reflector in the attic or on the gable is not possible, or if weather conditions raise concerns for a gable installation, the remaining option is to construct a standalone mast. At the top, a protective sphere should be installed to house the reflector, modem, and other necessary equipment. A sketch of the mast is shown in Fig. 23. The mast’s shaft, 6 – 8 meters high, can be made from sections of steel pipes with a diameter of 200 mm. A wooden mast made from a solid, thick trunk is also a suitable option. As for the foundation, decide based on your own experience and budget. One option could be using a well ring (see Fig. 23). A lightning rod must be installed at the top of the sphere.

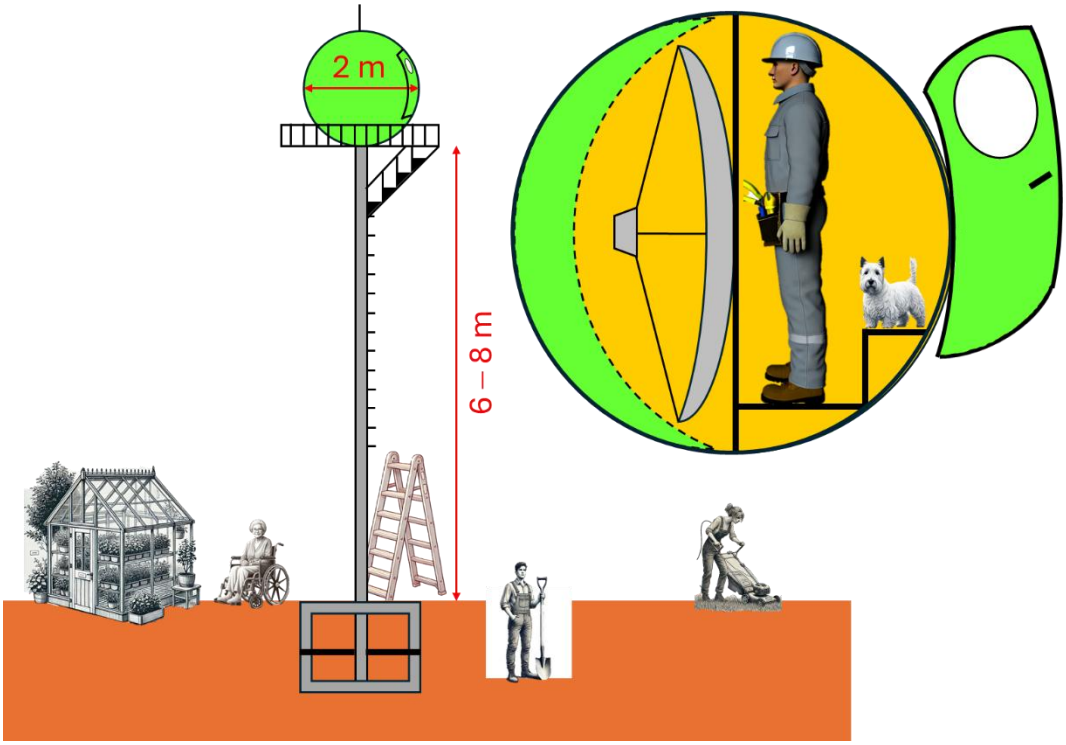


Fig. 23. Mast with a reflector inside a protective sphere.

The diameter of the sphere should slightly exceed the diameter of the reflector. It can be made from thin plywood petals, similar to the method used in reflector construction, and then coated with a water-repellent varnish to protect against weather and moisture. For additional protection, the surface of the sphere can be covered with fiberglass and epoxy resin, which, after curing, will form a strong protective layer. Alternatively, the plywood sphere can be used as a mold for producing a fully composite sphere via a molding process, resulting in a structure with high strength and low weight. The algorithm for cutting the petals is provided in the next section of the report.

In this report, we will limit ourselves to a brief discussion of issues related to station equipment. Its configuration will depend on your specific tasks and qualifications. Since the reflector will only occupy half of the internal space of the sphere, there is enough room to accommodate additional electronic equipment, aside from the modem. We plan to equip the reflector with an electromechanical system for precise alignment with the communication tower, as well as actuators that will allow the feed to rotate around its axis and adjust the focal length to optimize polarization and signal strength. It would also be beneficial to install an uninterruptible power supply (UPS). Inside the sphere, there will be a Raspberry Pi mini-computer connected to the modem via a LAN cable. This setup will enable Wi-Fi control of all peripheral devices used for automation on the site.

It is also necessary to consider a ventilation system inside the sphere, as it will obviously get very hot during the summer. The attic of a house also gets hot, but it is easier to ventilate. As for winter operation, further planning is required. For year-round operation of the electronic equipment, electric heating inside the sphere will be necessary, although the temperature requirements will be significantly lower than in living spaces. However, a far more significant problem could be humidity.

Petal cutting for sphere fabrication

Let us consider a quarter of a circle $z(x) = R - \sqrt{R^2 - x^2}$, with the radius R and $x \in [0, R]$, in the coordinate system shown in Fig. 24. By rotating this curve around the Z-axis, we obtain a hemisphere. The principle of cutting the hemisphere remains the same as for a paraboloid, but the equations will change. Additionally, in numerical calculations, it is necessary to resolve indeterminate forms like $0 \times \infty$, as explained below.

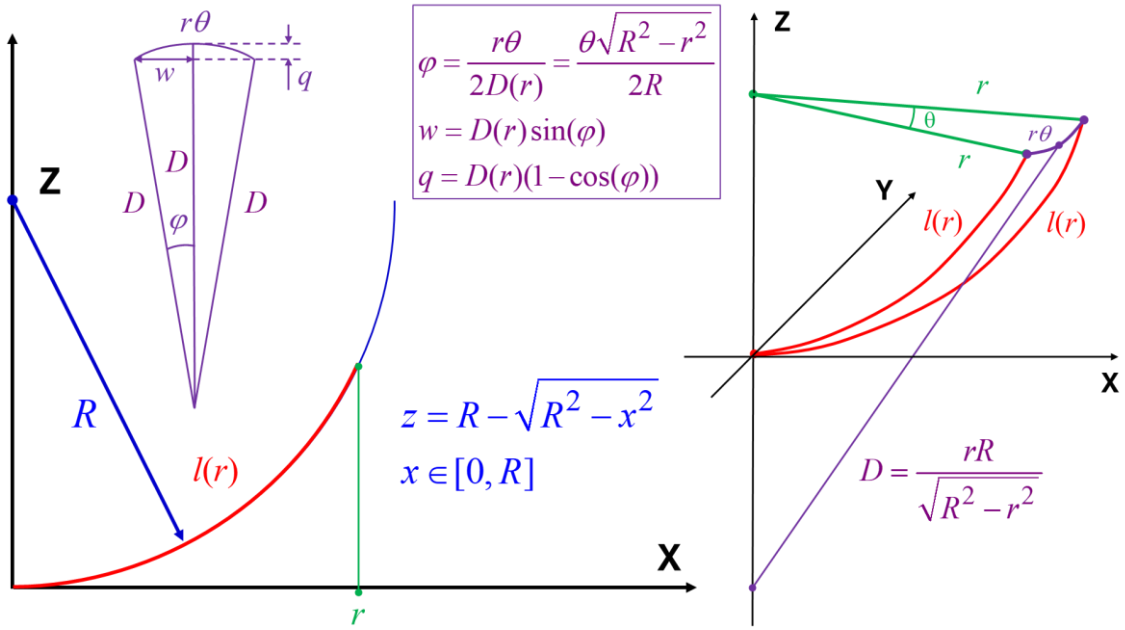


Fig. 24. Geometric parameters for calculating the petals from which the sphere will be cut.

Each radius $r \in [0, R]$, measured from the axis of rotation Z, will correspond to a specific circumference length $l(r)$:

$$l(r) = R\sin^{-1}\left(\frac{r}{R}\right) \tag{15}$$

Given the overall diameter of the sphere $D_s = 2R$, we obtain the length L of each petal as:

$$L = l(R) = \frac{\pi R}{2} \tag{16}$$

To determine the profile of the petal, we will need to calculate the value of r based on a given l along the circumference:

$$r = R\sin\left(\frac{l}{R}\right) \tag{17}$$

Compared to a parabola, we will not need to solve a nonlinear equation here. For the value of the tangent segment D, see Fig. 24, we obtain:

$$D = R\tan\left(\sin^{-1}\left(\frac{r}{R}\right)\right) = \frac{rR}{\sqrt{R^2 - r^2}} \tag{18}$$

Thus, D becomes infinite at the equator of the sphere, when $r = R$. In the case of a parabola, this did not occur at any finite r. Because of this, it is necessary to slightly modify the algorithm to avoid the separate calculation

of D, which can take arbitrarily large values and lead to arithmetic overflow during numerical calculations. Additionally, it is necessary to eliminate indeterminate forms like $0 \times \infty$, hat arise in the formulas for w and q. To achieve this, we will use the following Taylor series expansions as $\varphi \to 0$:

$$\sin(\varphi) = \varphi + O(\varphi^2) \tag{19}$$

$$\cos(\varphi) = 1 - \frac{\varphi^2}{2} + O(\varphi^4) \tag{20}$$

Taking into account (19) and (20), we can write the equations for w and q in the following equivalent form:

$$w = D(r)\sin(\varphi) = D(r)(\sin(\varphi) - \varphi) + D(r)\varphi \tag{21}$$

$$q = D(r)(1 - \cos(\varphi)) = D(r)\left(1 - \cos(\varphi) - \frac{\varphi^2}{2}\right) + D(r)\frac{\varphi^2}{2} \tag{22}$$

Using eq. (18) and $\varphi = \frac{\theta\sqrt{R^2 - r^2}}{2R}$ (see Fig. 24), we obtain:

$$w = \frac{r(\sin(\varphi) - \varphi)}{\sqrt{R^2 - r^2}} + \frac{r\theta}{2} \tag{23}$$

$$q = \frac{r\left(1 - \cos(\varphi) - \frac{\varphi^2}{2}\right)}{\sqrt{R^2 - r^2}} + \frac{r\theta^2\sqrt{R^2 - r^2}}{8R} \tag{24}$$

In (23) and (24), uncertainties no longer arise, as $(\sin(\varphi) - \varphi)$ and $\left(1 - \cos(\varphi) - \frac{\varphi^2}{2}\right)$ in the numerators decay much faster as $r \to R$ than the denominator $\sqrt{R^2 - r^2}$. Therefore, there will be no arithmetic overflow when calculating these ratios.

We are now ready to present the algorithm for cutting the sphere:

1. Calculate the angular width of the petal $\theta = 2\pi/N$, where N is the number of petals.
2. For a given R, calculate the total length of the petal L using eq. (16).
3. Calculate the array of petal lengths $l_i = iL/(M-1)$, $i = \overline{0, M-1}$, where M a sufficient number of divisions, determining the accuracy of the profile construction. Numbering starts from $i = 0$, as is customary in programming.
4. Calculate the array of radii r_i, corresponding to l_i using eq. (17), $i = \overline{0, M-1}$.
5. Calculate the array of opening angles of the petals $\varphi_i = \frac{\theta\sqrt{R^2 - r_i^2}}{2R}$, $i = \overline{0, M-1}$.
6. Calculate the array of displacements q_i using eq. (24), $i = \overline{0, M-1}$.
7. Calculate the array of meridional coordinates $s_i = l_i - q_i$, $i = \overline{0, M-1}$.
8. Calculate the array of widths w_i using eq. (23), $i = \overline{0, M-1}$, measured from the corresponding meridional coordinates s_i in both directions.
9. To close the petal, connect the ends of the final widths w_{M-1} with a straight line (infinite radius of curvature for the closing arc).

The Python code for calculating the profile of a sphere petal is provided below. It can be downloaded from the project repository (file Sphere.py). The program is a console application, meaning it does not have a user interface. All parameters (Design parameters) need to be entered directly into the program’s text. As a result of the program's execution, a tabular file named Spherical_profile_data.csv is created, containing columns with all the discrete parameters calculated by the algorithm. To plot the petal profile (s_i, w_i), only the last two columns, s and w, are required.

```
#
# Algorithm for cutting the petals for a sphere
#
# Dmitriy Makhnovskiy, September 2024
#

import csv
import numpy as np

pi = np.pi  # pi-constant 3.1415....

# Design parameters
R = 1000.0  # dish radius; your units (mm, cm, or m)
N = 10  # number of petals used for the sphere
M = 50  # number of points on the petal template for drawing its profile

# Calculated parameters
theta = 2.0 * pi / N  # angular width of the petal
L = pi * R / 2.0  # petal length

l = [0.0] * M  # points along the petal
l[M - 1] = L
r = [0.0] * M  # array of r corresponding to the array of l
r[M - 1] = R
eff = [0.0] * M  # array of the opening angles
eff[M - 1] = 0.0
q = [0.0] * M  # meridian displacements with respect to the l-points
q[M - 1] = 0.0
s = [0.0] * M  # array of the meridian coordinates
s[M - 1] = L
w = [0.0] * M  # array of the widths corresponding to the array of s
w[M - 1] = theta * R / 2.0

for i in range(0, M-1):
    length = i * L / (M - 1)
    l[i] = length
    r[i] = R * np.sin(length / R)
    sqroot = np.sqrt(R ** 2 - r[i] ** 2)
    eff[i] = theta * sqroot / (2.0 * R)
    q[i] = r[i] * (1.0 - np.cos(eff[i]) - eff[i] ** 2 / 2.0) / sqroot + r[i] * theta ** 2 * sqroot / (8.0 *
R)
    s[i] = l[i] - q[i]
    w[i] = r[i] * (np.sin(eff[i]) - eff[i]) / sqroot + r[i] * theta / 2.0

# Saving the profile data to a CSV file
data = np.column_stack((l, r, eff, q, s, w))
header = ['l', 'r', 'eff', 'q', 's', 'w']

with open('Spherical_profile_data.csv', 'w', newline='') as csv_file:
    writer = csv.writer(csv_file)
    writer.writerow(header)
    writer.writerows(data)
```

An example of a calculated petal for the sphere is shown in Fig. 25. As we know well from the school curriculum, this same principle is used in the creation of geographic atlases. A globe is divided into vertical segments called petals. These petals take the form of elongated slices (see Fig. 25) that span the surface from one pole to the other. Each slice represents a part of the globe's surface, which is then mapped onto a flat sheet. Each petal of the globe is projected onto a flat sheet of paper or map. Since a spherical surface cannot be perfectly flattened without distortions, special projections (such as gnomonic, sinusoidal, or Mercator projections) are used to minimize distortions of shape, area, and distance. Once all the globe's petals are unwrapped and projected, they are combined on the pages of the atlas.

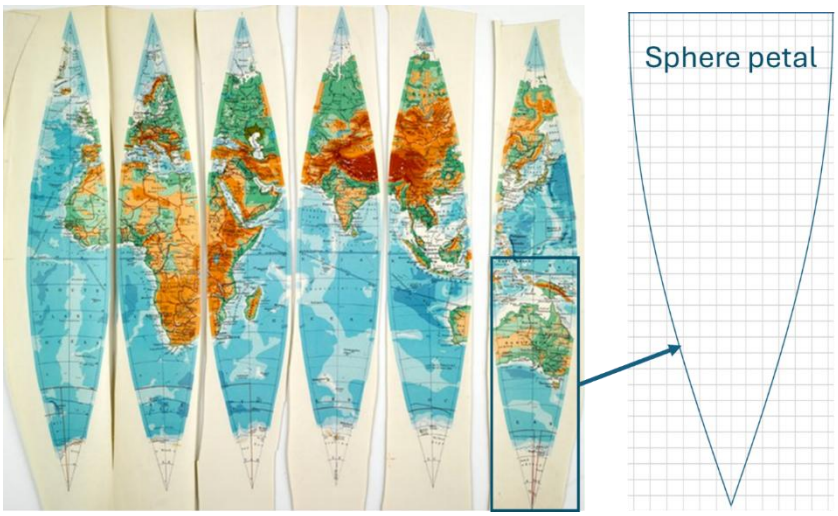


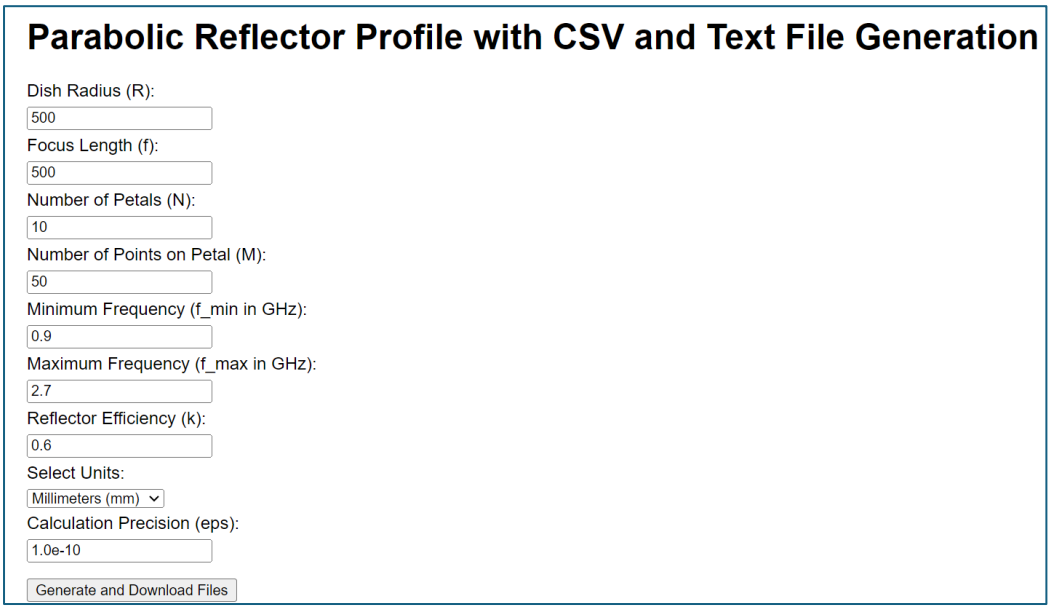
Fig. 25. Calculated sphere petal and the application of spherical mapping in cartography.

The assembly of the hemispherical petals must be performed with their seams fitting tightly against the rigid ribs of the frame, which replicate quarter circles. Only in this way can tangential joining of the petals be ensured, as described for the paraboloid. The frame can be positioned either outside or inside the surface. It will be easier to fasten the petals to the frame if it is located on the outside. The finished surface can be used repeatedly to create composite molds using the impression method.

Cutting algorithms with a graphical interface in the browser

Based on our Python code (Sphere.py and Parabolic_reflector.py), we have generated (using ChatGPT-4) JavaScript code with a graphical interface, accessible via a browser. These are saved in the files Sphere.html and Parabolic_reflector.html, and are available for download in the project repository. To run the programs, simply click on the file and allow your browser (Google Chrome, Microsoft Edge) to open the webpage. On the webpage, you will need to input the parameters for the paraboloid or sphere. To avoid errors, switch to the English keyboard when entering parameters. Since frequencies may be decimal numbers (e.g., 0.9 GHz), use a period (.) to input non-integer values. Please note that the programs will only work on a computer, not on a smartphone.

The use of JavaScript was made possible due to the absence of specific numerical libraries in our Python programs, thanks to the simplicity of the algorithms. This eliminates the need to install bulky Python environments and IDEs to run the calculations. The browser interfaces and the program code are shown below.



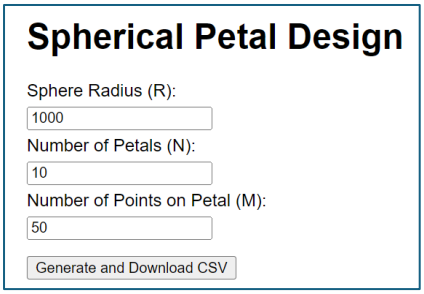
Parabolic_reflector.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Parabolic Reflector Profile</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        label, input, select {
            margin: 5px 0;
            display: block;
        }
        button {
            margin: 10px 0;
        }
        #output {
            margin-top: 20px;
            background-color: #f0f0f0;
            padding: 10px;
            border: 1px solid #ccc;
        }
    </style>
</head>
<body>
    <h1>Parabolic Reflector Profile with CSV and Text File Generation</h1>

    <label for="R">Dish Radius (R):</label>
    <input type="number" id="R" placeholder="Enter R">

    <label for="f">Focus Length (f):</label>
    <input type="number" id="f" placeholder="Enter f">

    <label for="N">Number of Petals (N):</label>
    <input type="number" id="N" placeholder="Enter N">

    <label for="M">Number of Points on Petal (M):</label>
```

```
    <input type="number" id="M" placeholder="Enter M">

    <label for="f_min">Minimum Frequency (f_min in GHz):</label>
    <input type="number" id="f_min" placeholder="Enter f_min">

    <label for="f_max">Maximum Frequency (f_max in GHz):</label>
    <input type="number" id="f_max" placeholder="Enter f_max">

    <label for="k">Reflector Efficiency (k):</label>
    <input type="number" id="k" step="0.01" placeholder="Enter efficiency (k)" value="0.6">

    <!-- Select units -->
    <label for="units">Select Units:</label>
    <select id="units">
        <option value="mm">Millimeters (mm)</option>
        <option value="cm">Centimeters (cm)</option>
        <option value="m">Meters (m)</option>
    </select>

    <!-- Input for calculation precision (eps) -->
    <label for="eps">Calculation Precision (eps):</label>
    <input type="number" id="eps" placeholder="Enter eps" step="1e-12" value="1.0e-10">

    <button id="generateBtn">Generate and Download Files</button>

    <!-- Div for displaying output -->
    <div id="output">The results will appear here...</div>

    <script>
        document.getElementById('generateBtn').addEventListener('click', function () {
            try {
                console.clear();
                console.log("Button clicked, starting process...");

                // Clear old output
                document.getElementById('output').innerHTML = "Processing...";

                // Get input values
                const R = parseFloat(document.getElementById('R').value);
                const f = parseFloat(document.getElementById('f').value);
                const N = parseInt(document.getElementById('N').value);
                const M = parseInt(document.getElementById('M').value);
                const f_min = parseFloat(document.getElementById('f_min').value) * 1e9; // GHz to Hz
                const f_max = parseFloat(document.getElementById('f_max').value) * 1e9; // GHz to Hz
                const k = parseFloat(document.getElementById('k').value);
                const units = document.getElementById('units').value;
                const eps = parseFloat(document.getElementById('eps').value); // Precision for calculation

                console.log("Input values received...");

                // Validate input
                if (isNaN(R) || isNaN(f) || isNaN(N) || isNaN(M) || isNaN(f_min) || isNaN(f_max) ||
isNaN(k) || isNaN(eps)) {
                    alert("Please enter valid values for all fields.");
                    return;
                }

                // Basic calculations (following Python code closely)
                const pi = Math.PI;
                const c = 2.99792458e8; // Speed of light in m/s
                const lambda_max = c / f_min;
                const lambda_min = c / f_max;

                let radius;
                if (units === 'mm') {
                    radius = R / 1000.0;
                } else if (units === 'cm') {
                    radius = R / 100.0;
                } else {
                    radius = R;
                }

```

```
                const Gain_min = 10.0 * Math.log10(k * (2.0 * pi * radius / lambda_max) ** 2);
                const Gain_max = 10.0 * Math.log10(k * (2.0 * pi * radius / lambda_min) ** 2);

                // Parabolic coefficient calculation
                const a = 1.0 / (4.0 * f);
                const theta = 2.0 * pi / N;
                const value = Math.sqrt(1.0 + 4.0 * a ** 2 * R ** 2);
                const L = (R * value / 2.0) + Math.log(2.0 * a * R + value) * f;

                // Calculating radius r using method of "dividing by half"
                function rl(length) {
                    let left = 0.0;
                    let right = R;
                    let r = (left + right) / 2.0;
                    let value = Math.sqrt(1.0 + 4.0 * a ** 2 * r ** 2);
                    value = r * value / 2.0 + Math.log(2.0 * a * r + value) * f - length;
                    let i = 1;
                    while (Math.abs(value) > eps && i < 100) {
                        if (value < 0.0) {
                            left = r;
                        } else if (value > 0.0) {
                            right = r;
                        }
                        r = (left + right) / 2.0;
                        value = Math.sqrt(1.0 + 4.0 * a ** 2 * r ** 2);
                        value = r * value / 2.0 + Math.log(2.0 * a * r + value) * f - length;
                        i++;
                    }
                    return r;
                }

                // Initialize arrays
                const l = new Array(M).fill(0);
                const r = new Array(M).fill(0);
                const D = new Array(M).fill(0);
                const eff = new Array(M).fill(0);
                const q = new Array(M).fill(0);
                const s = new Array(M).fill(0);
                const w = new Array(M).fill(0);

                l[M - 1] = L;
                r[M - 1] = R;
                D[M - 1] = R * Math.sqrt(1.0 + 4.0 * a ** 2 * R ** 2);
                eff[0] = theta / 2.0;
                eff[M - 1] = theta / (2.0 * Math.sqrt(1.0 + 4.0 * a ** 2 * R ** 2));
                q[M - 1] = D[M - 1] * (1.0 - Math.cos(eff[M - 1]));
                s[M - 1] = l[M - 1] - q[M - 1];
                w[M - 1] = D[M - 1] * Math.sin(eff[M - 1]);

                // Perform the calculations for each point
                for (let i = 1; i < M - 1; i++) {
                    const length = i * L / (M - 1);
                    l[i] = length;
                    r[i] = rl(length);
                    D[i] = r[i] * Math.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2);
                    eff[i] = theta / (2.0 * Math.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2));
                    q[i] = D[i] * (1.0 - Math.cos(eff[i]));
                    s[i] = l[i] - q[i];
                    w[i] = D[i] * Math.sin(eff[i]);
                }

                // Generate CSV content
                let csvContent = "l,r,D,eff,q,s,w\n";
                for (let i = 0; i < M; i++) {
                    csvContent +=
`${l[i].toFixed(4)},${r[i].toFixed(4)},${D[i].toFixed(4)},${eff[i].toFixed(4)},${q[i].toFixed(4)},${s[i].toFixe
d(4)},${w[i].toFixed(4)}\n`;
                }

                // Create and download CSV file
                const blobCsv = new Blob([csvContent], { type: "text/csv" });
                const linkCsv = document.createElement("a");
```

```
                linkCsv.href = URL.createObjectURL(blobCsv);
                linkCsv.download = "Parabolic_profile_data.csv";
                document.body.appendChild(linkCsv);
                linkCsv.click();
                document.body.removeChild(linkCsv);

                console.log("CSV file generated.");

                // Generate text file with design parameters
                let designParameters = '';
                designParameters += `Dish radius = ${R} ${units}\n`;
                designParameters += `Dish height = ${(a * R ** 2).toFixed(4)} ${units}\n`;
                designParameters += `Focus length of the reflector = ${f} ${units}\n`;
                designParameters += `Petal length = ${L.toFixed(4)} ${units}\n`;
                designParameters += `Number of petals = ${N}\n`;
                designParameters += `Petal radius of curvature = ${D[M - 1].toFixed(4)} ${units}\n`;
                designParameters += `Minimum frequency = ${(f_min / 1.0e9).toFixed(4)} GHz\n`;
                designParameters += `Maximum frequency = ${(f_max / 1.0e9).toFixed(4)} GHz\n`;
                designParameters += `Minimum wavelength = ${lambda_min.toFixed(4)} m\n`;
                designParameters += `Maximum wavelength = ${lambda_max.toFixed(4)} m\n`;
                designParameters += `Reflector efficiency = ${k}\n`;
                designParameters += `Minimum antenna gain = ${Gain_min.toFixed(4)} dB\n`;
                designParameters += `Maximum antenna gain = ${Gain_max.toFixed(4)} dB\n`;

                // Create and download text file
                const blobTxt = new Blob([designParameters], { type: "text/plain" });
                const linkTxt = document.createElement("a");
                linkTxt.href = URL.createObjectURL(blobTxt);
                linkTxt.download = "Design_parameters.txt";
                document.body.appendChild(linkTxt);
                linkTxt.click();
                document.body.removeChild(linkTxt);

                console.log("Text file with design parameters generated.");

                // Output result
                document.getElementById('output').innerHTML = "<h3>CSV and Text files generated and
downloaded.</h3>";

            } catch (error) {
                console.error("Error encountered:", error);
                alert("An error occurred. Check the console for more information.");
            }
        });
    </script>
</body>
</html>
```



Sphere.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Spherical Petal Design</title>
    <style>
        body {
```

```
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        label, input {
            margin: 5px 0;
            display: block;
        }
        button {
            margin: 10px 0;
        }
    </style>
</head>
<body>
    <h1>Spherical Petal Design</h1>

    <label for="R">Dish Radius (R):</label>
    <input type="number" id="R" placeholder="Enter Dish Radius (R)" step="0.1">

    <label for="N">Number of Petals (N):</label>
    <input type="number" id="N" placeholder="Enter Number of Petals (N)" step="1">

    <label for="M">Number of Points on Petal (M):</label>
    <input type="number" id="M" placeholder="Enter Number of Points (M)" step="1">

    <button id="downloadBtn">Generate and Download CSV</button>

    <script>
        document.getElementById('downloadBtn').addEventListener('click', function() {
            // Get input values
            const R = parseFloat(document.getElementById('R').value);
            const N = parseInt(document.getElementById('N').value);
            const M = parseInt(document.getElementById('M').value);

            // Check if any of the inputs are empty or invalid
            if (isNaN(R) || isNaN(N) || isNaN(M) || R <= 0 || N <= 0 || M <= 0) {
                alert('Please enter valid positive values for R, N, and M.');
                return; // Do nothing if values are invalid
            }

            // Calculate angular width and petal length
            const pi = Math.PI;
            const theta = 2.0 * pi / N;  // angular width of the petal
            const L = pi * R / 2.0;      // petal length

            // Initialize arrays
            const l = new Array(M).fill(0);
            const r = new Array(M).fill(0);
            const eff = new Array(M).fill(0);
            const q = new Array(M).fill(0);
            const s = new Array(M).fill(0);
            const w = new Array(M).fill(0);

            // Set the last points
            l[M - 1] = L;
            r[M - 1] = R;
            s[M - 1] = L;
            w[M - 1] = theta * R / 2.0;

            // Loop through the points and calculate the values
            for (let i = 0; i < M - 1; i++) {
                const length = i * L / (M - 1);
                l[i] = length;
                r[i] = R * Math.sin(length / R);
                const sqroot = Math.sqrt(R ** 2 - r[i] ** 2);
                eff[i] = theta * sqroot / (2.0 * R);
                q[i] = r[i] * (1.0 - Math.cos(eff[i]) - eff[i] ** 2 / 2.0) / sqroot +
                       r[i] * theta ** 2 * sqroot / (8.0 * R);
                s[i] = l[i] - q[i];
                w[i] = r[i] * (Math.sin(eff[i]) - eff[i]) / sqroot + r[i] * theta / 2.0;
            }

            // Prepare the CSV data
```

```
            let csvContent = 'l, r, eff, q, s, w\n';
            for (let i = 0; i < M; i++) {
                csvContent += `${l[i].toFixed(2)}, ${r[i].toFixed(2)}, ${eff[i].toFixed(4)},
${q[i].toFixed(4)}, ${s[i].toFixed(2)}, ${w[i].toFixed(2)}\n`;
            }

            // Create a Blob from the CSV data
            const blob = new Blob([csvContent], { type: 'text/csv' });
            const url = URL.createObjectURL(blob);

            // Create a temporary link and trigger download
            const a = document.createElement('a');
            a.href = url;
            a.download = 'Spherical_profile_data.csv';
            document.body.appendChild(a);
            a.click();
            document.body.removeChild(a);
            URL.revokeObjectURL(url);  // Free up memory
        });
    </script>
</body>
</html>
```

Creation of electronic files for laser cutting of petals

At this stage of the project, we have not used laser cutting, so we do not have the relevant experience. However, we decided to include this section to demonstrate how it will be carried out at the next stages.

The diameter of the laser beam in cutters typically varies depending on the type of laser and focusing system. Here are some approximate values for common types of lasers:

CO2 lasers

- The beam diameter usually ranges from 0.1 mm to 0.3 mm.
- This is one of the most commonly used types of lasers for cutting thin materials, such as plywood with a thickness of $3 - 6$ mm.

Fibre lasers

- The beam diameter is usually smaller than that of CO2 lasers and can range from 0.05 mm to 0.1 mm.
- These lasers are used less frequently for plywood but provide more precise cutting when needed.

Diode lasers

- The beam diameter can range from 0.1 mm to 0.5 mm, depending on the focusing.
- These are more affordable laser options for small home machines, but their power is lower compared to CO2 and fibre lasers.

Thus, for thin plywood, a CO2 laser with a beam diameter of approximately $0.1 - 0.3$ mm is most commonly used. The choice of beam diameter may depend on the cutting precision requirements and the cutter's configuration settings.

The laser will be used to cut both the petals of the reflector and sphere, as well as the frames supporting them. For now, we are only discussing the cutting of the petals, as the frames have not yet been designed. After calculating the petal profile, it is necessary to create a special electronic file for laser cutting. Laser cutters use file formats supported by CNC systems, such as DXF, G-code, and SVG.

Let us first consider cutting a spherical petal, as it is simpler. A JavaScript algorithm will be created to generate G-code for cutting the petal and save it in the appropriate file format. The algorithm works in the following sequence:

Reading the profile from a CSV file

The calculated profile data was saved in Sphere_profile_data.csv in columns 5 and 6: s_i (X-coordinate) and w_i (Y-coordinate). These coordinates are read and interpreted as points along the path of the laser beam. For laser cutting, it is necessary to create a profile with a large number of points (M) to accurately reproduce the shape. Since the cutting is automated, hundreds of points can be chosen.

Laser diameter compensation

To compensate for the width of the cut kerf, half of the laser beam diameter will be added to the Y-coordinates of the petal.

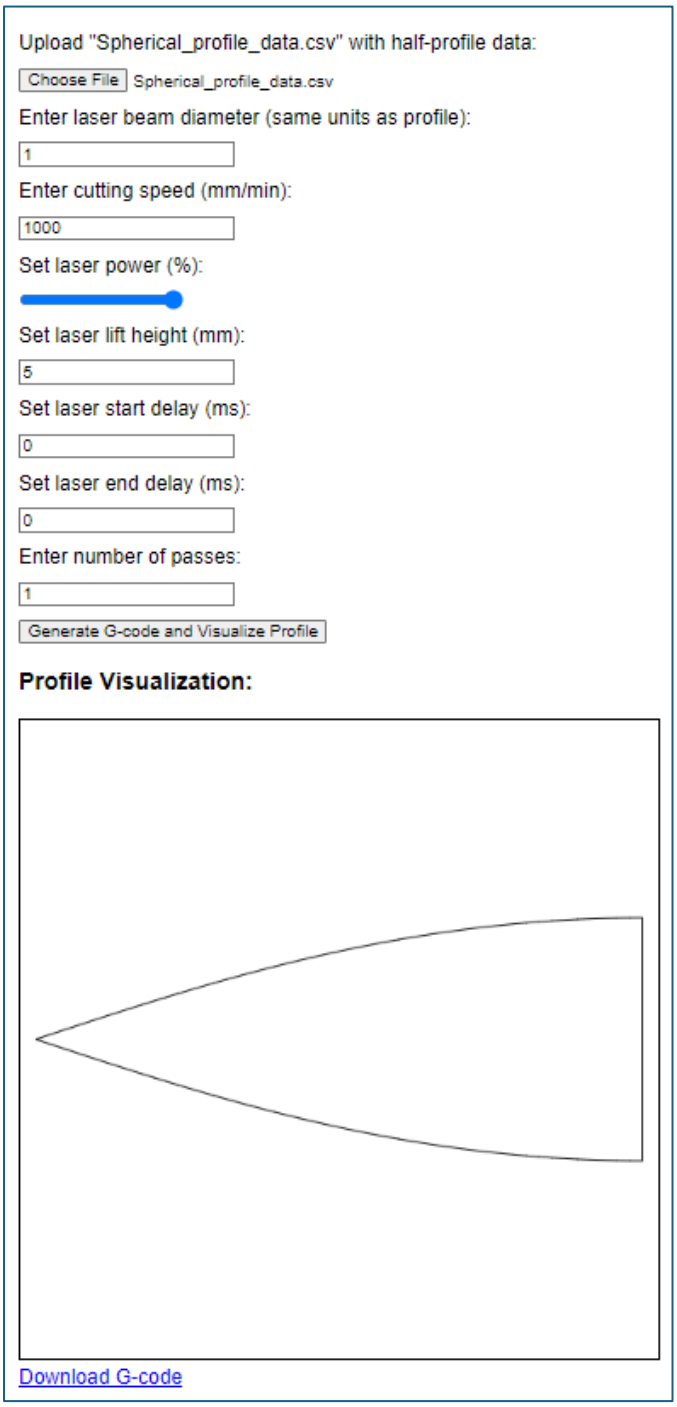
Creating the complete profile

Since the calculated profile in Sphere_profile_data.csv is symmetric relative to the X-axis, it will be mirrored to create the second half of the petal. The resulting two branches of the profile originate from a single point. Their free ends need to be connected by a straight line (forming the petal for the sphere). As a result, a closed contour of the petal will be created, ready for cutting. For moving the laser beam along the straight line connecting the two points, no additional points are required, which simplifies the code.

G-code generation

For each point of the profile, a laser movement command (G1 command) is generated with the exact X and Y coordinates. Laser on and off commands (G1 Z0 for starting the cut and G0 Z5 for raising the laser) are added to properly control the cutting process.

To create the G-code file, we generated a JavaScript code saved in the project repository, which is shown below along with the browser interface. To check the petal shape in the interface window, click on "Generate G-code and Visualize Profile". If everything is correct, click on "Download G-code" to save the G-code file Full_sphere_profile_gcode.nc.



Example settings for the GUI:

- Cutting speed — Numeric input field in mm/min.
- Laser power — Slider for adjusting from 0% to 100%.
- Laser lift height — Input field for height values (e.g., 5 mm).
- Laser on/off delay — Input fields for delay in milliseconds before starting the cut and after finishing (e.g., 100 ms).
- Number of passes — Input field for specifying the number of passes (e.g., 1, 2, 3...).

Cutting_sphere_petal.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Laser Cutting Profile Generator</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        label, input, button {
            display: block;
            margin: 10px 0;
        }
        canvas {
            border: 1px solid black;
        }
    </style>
</head>
<body>

    <h1>Generate G-code for Laser Cutting with Profile Visualization</h1>

    <label for="csvFile">Upload "Spherical_profile_data.csv" with half-profile data:</label>
    <input type="file" id="csvFile" accept=".csv">

    <label for="laserDiameter">Enter laser beam diameter (same units as profile):</label>
    <input type="number" id="laserDiameter" placeholder="Enter beam diameter">

    <label for="feedRate">Enter cutting speed (mm/min):</label>
    <input type="number" id="feedRate" placeholder="Enter speed (default: 1000)" value="1000">

    <label for="laserPower">Set laser power (%):</label>
    <input type="range" id="laserPower" min="0" max="100" value="100">

    <label for="laserHeight">Set laser lift height (mm):</label>
    <input type="number" id="laserHeight" placeholder="Enter height (default: 5 mm)" value="5">

    <label for="laserDelayStart">Set laser start delay (ms):</label>
    <input type="number" id="laserDelayStart" placeholder="Enter delay (default: 0 ms)" value="0">

    <label for="laserDelayEnd">Set laser end delay (ms):</label>
    <input type="number" id="laserDelayEnd" placeholder="Enter delay (default: 0 ms)" value="0">

    <label for="laserPasses">Enter number of passes:</label>
    <input type="number" id="laserPasses" placeholder="Enter number of passes (default: 1)" value="1">

    <button id="generateGCode">Generate G-code and Visualize Profile</button>

    <h3>Profile Visualization:</h3>
    <canvas id="profileCanvas" width="500" height="500"></canvas>

    <!-- Placeholder for download link -->
    <div id="downloadLink"></div>

    <script>
        document.getElementById('generateGCode').addEventListener('click', function() {
            const csvFile = document.getElementById('csvFile').files[0];
            const laserDiameter = parseFloat(document.getElementById('laserDiameter').value);
            const feedRate = parseFloat(document.getElementById('feedRate').value);
            const laserPower = parseFloat(document.getElementById('laserPower').value);
            const laserHeight = parseFloat(document.getElementById('laserHeight').value);
            const laserDelayStart = parseFloat(document.getElementById('laserDelayStart').value);
            const laserDelayEnd = parseFloat(document.getElementById('laserDelayEnd').value);
            const laserPasses = parseInt(document.getElementById('laserPasses').value);

            if (!csvFile) {
                alert('Please upload the "Spherical_profile_data.csv" file.');
                return;
```

```
71.             }
72.
73.             if (isNaN(laserDiameter) || laserDiameter <= 0) {
74.                 alert('Please enter a valid laser beam diameter.');
75.                 return;
76.             }
77.
78.             if (isNaN(feedRate) || feedRate <= 0) {
79.                 alert('Please enter a valid feed rate.');
80.                 return;
81.             }
82.
83.             if (isNaN(laserPasses) || laserPasses <= 0) {
84.                 alert('Please enter a valid number of passes.');
85.                 return;
86.             }
87.
88.             const reader = new FileReader();
89.             reader.onload = function(event) {
90.                 const csvData = event.target.result;
91.                 const lines = csvData.split('\n');
92.                 const xCoords = [];  // 5th column (X axis)
93.                 const yCoords = [];  // 6th column (Y axis)
94.
95.                 // Parse the CSV data, skipping the header (first line)
96.                 for (let i = 1; i < lines.length; i++) {  // Skip the header row
97.                     const columns = lines[i].split(',');
98.                     if (columns.length >= 6) {  // Ensure there are enough columns
99.                         const x = parseFloat(columns[4]);  // 5th column (index 4)
100.                        const y = parseFloat(columns[5]);  // 6th column (index 5)
101.
102.                        if (!isNaN(x) && !isNaN(y)) {
103.                            xCoords.push(x);
104.                            yCoords.push(y + laserDiameter / 2);  // Compensate laser burn by adding half
diameter
105.                        }
106.                    }
107.                }
108.
109.                // Check if data is valid
110.                if (xCoords.length === 0 || yCoords.length === 0) {
111.                    alert('No valid data found in the file.');
112.                    return;
113.                }
114.
115.                // Create arrays for the full profile (including mirrored points)
116.                const fullXCoords = [...xCoords];
117.                const fullYCoords = [...yCoords];
118.
119.                // Mirror the profile by reflecting relative to X-axis (negate Y coordinates)
120.                for (let i = xCoords.length - 1; i >= 0; i--) {
121.                    fullXCoords.push(xCoords[i]);  // Keep X the same
122.                    fullYCoords.push(-yCoords[i]); // Reflect Y (negate Y)
123.                }
124.
125.                // Connect the last point of the mirrored profile with the first point to close the
profile
126.                fullXCoords.push(fullXCoords[0]); // Closing the profile on X-axis
127.                fullYCoords.push(fullYCoords[0]); // Closing the profile on Y-axis
128.
129.                // Calculate scaling factors to fit the profile into the canvas
130.                const minX = Math.min(...fullXCoords);
131.                const maxX = Math.max(...fullXCoords);
132.                const minY = Math.min(...fullYCoords);
133.                const maxY = Math.max(...fullYCoords);
134.
135.                const rangeX = maxX - minX;
136.                const rangeY = maxY - minY;
137.
138.                const canvas = document.getElementById('profileCanvas');
139.                const ctx = canvas.getContext('2d');
140.
```

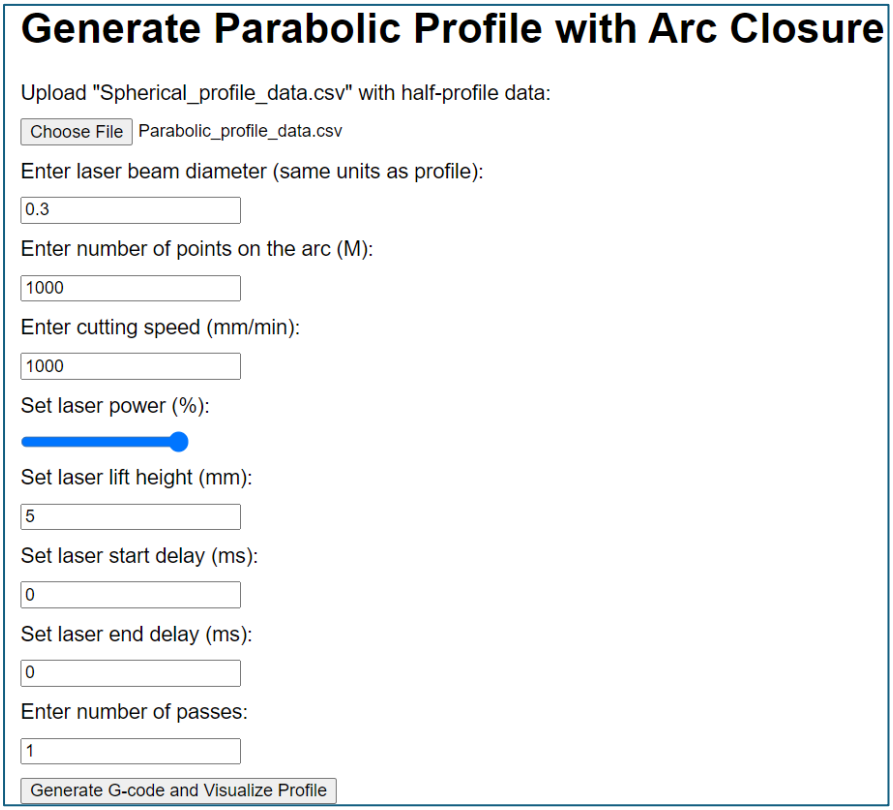
```
                // Clear the canvas
                ctx.clearRect(0, 0, canvas.width, canvas.height);

                // Scale the profile to fit into the canvas (maintaining aspect ratio)
                const scale = Math.min(canvas.width / rangeX, canvas.height / rangeY) * 0.95;  // Adjust
scale to fill canvas
                const offsetX = canvas.width / 2 - ((maxX + minX) / 2) * scale;
                const offsetY = canvas.height / 2 + ((maxY + minY) / 2) * scale;

                // Draw the profile on the canvas
                ctx.beginPath();
                ctx.moveTo(offsetX + fullXCoords[0] * scale, offsetY - fullYCoords[0] * scale); // Start
at the first point
                for (let i = 1; i < fullXCoords.length; i++) {
                    ctx.lineTo(offsetX + fullXCoords[i] * scale, offsetY - fullYCoords[i] * scale); //
Draw each segment
                }
                ctx.closePath(); // Close the path to complete the profile
                ctx.stroke(); // Outline the profile

                // Generate G-code
                let gcode = "G21 ; Set units to mm\n";
                gcode += `G90 ; Absolute positioning\n`;
                gcode += `S${laserPower} ; Set laser power to ${laserPower}%\n`;

                for (let pass = 0; pass < laserPasses; pass++) {
                    gcode += `G0 Z${laserHeight} ; Raise laser to ${laserHeight} mm\n`;
                    gcode += `G4 P${laserDelayStart} ; Start delay ${laserDelayStart} ms\n`;

                    gcode += `G0 X${fullXCoords[0].toFixed(4)} Y${fullYCoords[0].toFixed(4)} ; Move to
starting point\n`;
                    gcode += "G1 Z0 ; Lower laser for cutting\n";

                    for (let i = 0; i < fullXCoords.length; i++) {
                        gcode += `G1 X${fullXCoords[i].toFixed(4)} Y${fullYCoords[i].toFixed(4)}
F${feedRate} ; Cutting to (${fullXCoords[i].toFixed(4)}, ${fullYCoords[i].toFixed(4)})\n`;
                    }

                    gcode += `G4 P${laserDelayEnd} ; End delay ${laserDelayEnd} ms\n`;
                    gcode += "G0 Z5 ; Raise the laser\n"; // Finish pass
                }

                gcode += "M05 ; Turn off the laser\n";
                gcode += "M30 ; End of program\n";

                // Create a download link for the G-code file
                const blob = new Blob([gcode], { type: "text/plain" });
                const link = document.createElement('a');
                link.href = URL.createObjectURL(blob);
                link.download = 'Full_sphere_profile_gcode.nc';
                link.innerText = 'Download G-code';

                // Replace existing link (if any)
                const downloadLinkDiv = document.getElementById('downloadLink');
                downloadLinkDiv.innerHTML = '';  // Clear previous link
                downloadLinkDiv.appendChild(link);
            };

            reader.readAsText(csvFile);
        });
    </script>
</body>
</html>
```

Creating a closed petal profile for a paraboloid is a bit more complex, as the free ends of the profile branches are connected by an arc, not a straight line, as in the case of the sphere. Let's recall the structure of the Parabolic_profile_data.csv file, shown in Fig. 4. Assuming that the profile curves originate from the point with coordinates X = 0 and Y = 0, the JavaScript algorithm for the coordinates of the points on the closing arc will be as follows:

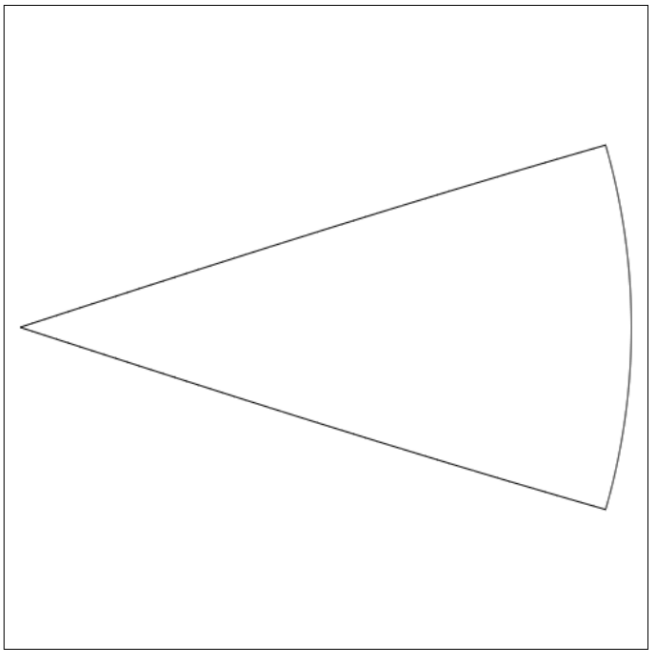
```
// Create arrays for the arc closure
const arcXCoords = [];
const arcYCoords = [];

// Algorithm to calculate arc points
for (let i = 0; i < M; i++) {
   const eff = effmax - (2.0 * effmax * i) / (M - 1);
   const arcX = Dmax * Math.cos(eff) + lmax - Dmax;
   const arcY = Dmax * Math.sin(eff);
   arcXCoords.push(arcX);
   arcYCoords.push(arcY);
}
```

where lmax is the petal length (the last value in the l-column of the Parabolic_profile_data.csv file), effmax is the maximum angle of the petal (the last value in the eff-column), and Dmax is the maximum length of the tangent segment (the last value in the D-column). Note that the value of Dmax has already been increased by half the laser beam diameter, although such precision is hardly necessary during the assembly of the reflector. The coordinates of the center of the arc (see Fig. 6) will be X = (lmax – Dmax) и Y = 0.

The JavaScript program code along with the browser interface are shown below. It can be downloaded from the project repository. To check the petal shape in the interface window, click on "Generate G-code and Visualize Profile". If everything is correct, click on "Download G-code" to save the G-code file Full_paraboloid_profile_gcode.nc.



Profile Visualization:



Download G-code

Cutting_paraboloid_petal.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Laser Cutting Profile with Arc Closure</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        label, input, button {
            display: block;
            margin: 10px 0;
        }
        canvas {
            border: 1px solid black;
        }
    </style>
</head>
<body>

    <h1>Generate Parabolic Profile with Arc Closure</h1>

    <label for="csvFile">Upload "Spherical_profile_data.csv" with half-profile data:</label>
    <input type="file" id="csvFile" accept=".csv">

    <label for="laserDiameter">Enter laser beam diameter (same units as profile):</label>
    <input type="number" id="laserDiameter" placeholder="Enter beam diameter">

    <label for="M">Enter number of points on the arc (M):</label>
    <input type="number" id="M" placeholder="Enter number of points on the arc" value="100">

    <label for="feedRate">Enter cutting speed (mm/min):</label>
    <input type="number" id="feedRate" placeholder="Enter speed (default: 1000)" value="1000">

    <label for="laserPower">Set laser power (%):</label>
    <input type="range" id="laserPower" min="0" max="100" value="100">

    <label for="laserHeight">Set laser lift height (mm):</label>
```

```
    <input type="number" id="laserHeight" placeholder="Enter height (default: 5 mm)" value="5">

    <label for="laserDelayStart">Set laser start delay (ms):</label>
    <input type="number" id="laserDelayStart" placeholder="Enter delay (default: 0 ms)" value="0">

    <label for="laserDelayEnd">Set laser end delay (ms):</label>
    <input type="number" id="laserDelayEnd" placeholder="Enter delay (default: 0 ms)" value="0">

    <label for="laserPasses">Enter number of passes:</label>
    <input type="number" id="laserPasses" placeholder="Enter number of passes (default: 1)" value="1">

    <button id="generateGCode">Generate G-code and Visualize Profile</button>

    <h3>Profile Visualization:</h3>
    <canvas id="profileCanvas" width="500" height="500"></canvas>

    <!-- Placeholder for download link -->
    <div id="downloadLink"></div>

    <script>
        document.getElementById('generateGCode').addEventListener('click', function() {
            const csvFile = document.getElementById('csvFile').files[0];
            const laserDiameter = parseFloat(document.getElementById('laserDiameter').value);
            const feedRate = parseFloat(document.getElementById('feedRate').value);
            const laserPower = parseFloat(document.getElementById('laserPower').value);
            const laserHeight = parseFloat(document.getElementById('laserHeight').value);
            const laserDelayStart = parseFloat(document.getElementById('laserDelayStart').value);
            const laserDelayEnd = parseFloat(document.getElementById('laserDelayEnd').value);
            const laserPasses = parseInt(document.getElementById('laserPasses').value);
            const M = parseInt(document.getElementById('M').value);  // Number of points on the arc

            if (!csvFile) {
                alert('Please upload the "Spherical_profile_data.csv" file.');
                return;
            }

            if (isNaN(laserDiameter) || laserDiameter <= 0) {
                alert('Please enter a valid laser beam diameter.');
                return;
            }

            if (isNaN(M) || M <= 10) {
                alert('Please enter a valid number of points (M > 10).');
                return;
            }

            const reader = new FileReader();
            reader.onload = function(event) {
                const csvData = event.target.result;
                const lines = csvData.split('\n');
                const xCoords = [];  // 6th column (X axis)
                const yCoords = [];  // 7th column (Y axis)
                let effmax, Dmax, lmax;

                // Parse the CSV data, skipping the header (first line)
                for (let i = 1; i < lines.length; i++) {  // Skip the header row
                    const columns = lines[i].split(',');
                    if (columns.length >= 7) {  // Ensure there are enough columns
                        const x = parseFloat(columns[5]);  // 6th column (index 5)
                        const y = parseFloat(columns[6]);  // 7th column (index 6)
                        const eff = parseFloat(columns[3]);  // 4th column (eff)
                        const D = parseFloat(columns[2]);  // 3rd column (D)
                        const l = parseFloat(columns[0]);  // 1st column (l)

                        if (!isNaN(x) && !isNaN(y) && !isNaN(eff)) {
                            xCoords.push(x);
                            yCoords.push(y + laserDiameter / 2);  // Compensate laser burn by adding half
diameter
                            lmax = l;  // Last value in the 1st column
                            Dmax = D + laserDiameter / 2;  // Last value in the 3rd column
                            effmax = eff;  // Last value in the 4th column
                        }
```

```
                    }
                }

                // Check if data is valid
                if (xCoords.length === 0 || yCoords.length === 0) {
                    alert('No valid data found in the file.');
                    return;
                }

                // Create arrays for the arc closure
                const arcXCoords = [];
                const arcYCoords = [];

                // Algorithm to calculate arc points
                for (let i = 0; i < M; i++) {
                    const eff = effmax - (2.0 * effmax * i) / (M - 1);
                    const arcX = Dmax * Math.cos(eff) + lmax - Dmax;
                    const arcY = Dmax * Math.sin(eff);
                    arcXCoords.push(arcX);
                    arcYCoords.push(arcY);
                }

                // Create arrays for the full profile (including arc and mirrored points)
                const fullXCoords = [...xCoords];
                const fullYCoords = [...yCoords];

                // Add arc points to close the profile
                fullXCoords.push(...arcXCoords);
                fullYCoords.push(...arcYCoords);

                // Mirror the profile by reflecting relative to X-axis (keep X the same and reverse Y)
                for (let i = xCoords.length - 1; i >= 0; i--) {
                    fullXCoords.push(xCoords[i]);  // X-axis stays the same
                    fullYCoords.push(-yCoords[i]); // Reflect Y relative to the X-axis
                }

                // Calculate scaling factors to fit the profile into the canvas
                const minX = Math.min(...fullXCoords);
                const maxX = Math.max(...fullXCoords);
                const minY = Math.min(...fullYCoords);
                const maxY = Math.max(...fullYCoords);

                const rangeX = maxX - minX;
                const rangeY = maxY - minY;

                const canvas = document.getElementById('profileCanvas');
                const ctx = canvas.getContext('2d');

                // Clear the canvas
                ctx.clearRect(0, 0, canvas.width, canvas.height);

                // Scale the profile to fit into the canvas (maintaining aspect ratio)
                const scale = Math.min(canvas.width / rangeX, canvas.height / rangeY) * 0.95;  // Adjust
scale to fill canvas
                const offsetX = canvas.width / 2 - ((maxX + minX) / 2) * scale;
                const offsetY = canvas.height / 2 + ((maxY + minY) / 2) * scale;

                // Draw the profile on the canvas
                ctx.beginPath();
                ctx.moveTo(offsetX + fullXCoords[0] * scale, offsetY - fullYCoords[0] * scale); // Start
at the first point
                for (let i = 1; i < fullXCoords.length; i++) {
                    ctx.lineTo(offsetX + fullXCoords[i] * scale, offsetY - fullYCoords[i] * scale); //
Draw each segment
                }
                ctx.closePath(); // Close the path to complete the profile
                ctx.stroke(); // Outline the profile

                // Generate G-code
                let gcode = "G21 ; Set units to mm\n";
                gcode += `G90 ; Absolute positioning\n`;
                gcode += `S${laserPower} ; Set laser power to ${laserPower}%\n`;
```

```
181.
182.                 for (let pass = 0; pass < laserPasses; pass++) {
183.                     gcode += `G0 Z${laserHeight} ; Raise laser to ${laserHeight} mm\n`;
184.                     gcode += `G4 P${laserDelayStart} ; Start delay ${laserDelayStart} ms\n`;
185.
186.                     gcode += `G0 X${fullXCoords[0].toFixed(4)} Y${fullYCoords[0].toFixed(4)} ; Move to
starting point\n`;
187.                     gcode += "G1 Z0 ; Lower laser for cutting\n";
188.
189.                     for (let i = 0; i < fullXCoords.length; i++) {
190.                         gcode += `G1 X${fullXCoords[i].toFixed(4)} Y${fullYCoords[i].toFixed(4)}
F${feedRate} ; Cutting to (${fullXCoords[i].toFixed(4)}, ${fullYCoords[i].toFixed(4)})\n`;
191.                     }
192.
193.                     gcode += `G4 P${laserDelayEnd} ; End delay ${laserDelayEnd} ms\n`;
194.                     gcode += "G0 Z5 ; Raise the laser\n"; // Finish pass
195.                 }
196.
197.                 gcode += "M05 ; Turn off the laser\n";
198.                 gcode += "M30 ; End of program\n";
199.
200.                 // Create a download link for the G-code file
201.                 const blob = new Blob([gcode], { type: "text/plain" });
202.                 const link = document.createElement('a');
203.                 link.href = URL.createObjectURL(blob);
204.                 link.download = 'Full_paraboloid_profile_gcode.nc';
205.                 link.innerText = 'Download G-code';
206.
207.                 // Replace existing link (if any)
208.                 const downloadLinkDiv = document.getElementById('downloadLink');
209.                 downloadLinkDiv.innerHTML = '';  // Clear previous link
210.                 downloadLinkDiv.appendChild(link);
211.             };
212.
213.             reader.readAsText(csvFile);
214.         });
215.     </script>
216. </body>
217. </html>
```

Conclusions

As a result of the initial stage of the project, algorithms for unfolding the petals of a paraboloid and a sphere, as well as algorithms for their laser cutting, were developed and implemented. The overall system layout was defined, and the key technological aspects of the reflector and protective sphere manufacturing process were elaborated. In addition, a method for pointing the reflector towards a communication mast was developed and tested. In the next stage of the project, further refinement, improvement, and automation of these methods are planned. Moreover, additional electromechanical devices will need to be developed to improve the precision of reflector alignment and adjust the polarization of the feeds. Besides utilizing commercially available wideband feeds, we are considering the development and testing of custom designs.

The question of building an access point mast (Fig. 23) is still open. An option to install the system in the attic of a house is being considered. However, a higher placement of the reflector and modem could significantly improve the quality of the Internet signal reception and transmission, as well as extend the Wi-Fi coverage area to a radius of up to 150– 200 meters. This would allow multiple neighbours to connect to the system for shared use. Additionally, the extended Wi-Fi coverage would enable the connection of

peripheral devices used for automation and control, such as greenhouses, irrigation systems, pump stations, surveillance cameras, and other equipment.

Recently, there has been information about the development of a new hybrid communication protocol that combines Wi-Fi with the Long Range (LoRa) networking protocol, called WiLo. This new wireless communication concept is aimed at utilizing existing Wi-Fi and LoRa hardware and is intended for Internet of Things (IoT) applications, such as sensor networks for agriculture and smart cities. Wi-Fi, as it stands today, is limited by its range and high-power consumption, whereas LoRa is characterized by low power requirements and long-range communication capabilities. The WiLo technology combines the advantages of both systems, allowing standard Wi-Fi devices to transmit data over long distances without additional hardware, thereby reducing costs, complexity, and potential points of failure. Experiments have shown that WiLo can operate effectively at distances of up to 500 meters with a success rate of 96%. One of the main advantages of WiLo is its ability to function on existing hardware without significant additional costs. However, a drawback remains the increased power consumption required for Wi-Fi devices to simultaneously handle communication and signal emulation. In the future, researchers plan to improve energy efficiency, data transmission rates, and system resilience to interference, as well as introduce security measures for cross-technology communication.

The use of composite materials for creating lightweight and durable structures appears to be highly promising. Implementing such technologies within the conditions of cottage workshops could represent a significant technological achievement.