

# Общие принципы построения точки доступа в интернет с использованием самодельного параболического рефлектора, 4G модема и облучателя

Дмитрий Махновский, Александр Махновский, и Людмила Махновская

Участки 32 и 62, СНТ «Рогачево», с. Трехсвятское, Дмитровский район, Московская область

Техническая консультация: [dmitriy.makhnovskiy@gmail.com](mailto:dmitriy.makhnovskiy@gmail.com) (Дмитрий)

Репозиторий проекта в открытом доступе: [https://github.com/DYK-Team/Parabolic\\_reflector](https://github.com/DYK-Team/Parabolic_reflector)

Пользуйтесь гиперссылками: кликнув на подчеркнутый текст, вы перейдете по ссылке на веб-страницу



## Содержание

Цели и задачи проекта .....	2
Раскройка лепестков для изготовления параболоида .....	3
Технология изготовления параболического рефлектора из лепестков .....	12
Монтаж рефлектора и его ориентация на вышку связи .....	17
Оснащения рефлектора 4G модемом и облучателем .....	23
Общие принципы построения мачтовой станции приема .....	37
Раскройка лепестков для изготовления сферы .....	39
Алгоритмы раскройки с графическим интерфейсом в браузере .....	43
Parabolic_reflector.html .....	44
Sphere.html.....	47
Создание электронных файлов для лазерной резки лепестков .....	49
Cutting_sphere_petal.html.....	52
Cutting_paraboloid_petal.html.....	56
Заключение .....	59

## Цели и задачи проекта

Изначальная цель этого проекта — обеспечить более или менее стабильный 4G интернет на даче. Универсального решения в данном случае не существует, так как условия приема сигнала сильно варьируются в зависимости от местности и могут меняться даже на небольших участках из-за различных препятствий, таких как здания, возвышенности, низины или лес. Эти факторы могут создавать сложную интерференционную картину, влияющую на уровень сигнала. Первый шаг — выбрать оператора связи с наилучшим покрытием 4G в вашем районе. Полезно также узнать опыт соседей, которые уже пытались организовать интернет, и расспросить их об используемом оборудовании. Однако уже на этом этапе можно столкнуться с противоречивыми отзывами, что создаст неопределенность. В такой ситуации лучше всего начать с технически оправданных и бюджетных решений.

*Наш проект предназначен для тех, кто, обладая любознательностью и свободным временем, стремится приобрести новые практические знания и навыки. Основная идея нашего подхода заключается в создании самодельного параболического рефлектора большого диаметра и его дооснащение MIMO или SISO облучателем для подключения к 4G модему. Нам удалось установить такой рефлектор на чердаке, что делает решение особенно экономичным. В дальнейшем проект может развиваться в направлении создания более сложной станции приема с отдельно стоящей мачтой, на вершине которой будет размещен параболический рефлектор в сферическом защитном кожухе. Внутри кожуха также планируется разместить все необходимое оборудование. Безусловно, это будет гораздо более дорогостоящая система, однако ее создание практически оправдано, если она будет обеспечивать интернетом несколько участков. Образовательный аспект проекта на всех его этапах, как текущем, так и последующих, играет для нас ключевую роль и служит важным источником мотивации. В условиях стремительного развития телекоммуникационных и компьютерных технологий нам хотелось бы разобраться в этих областях.*

В представленном отчете, охватывающем преимущественно конструкционные аспекты проекта, мы рассматриваем следующие методы:

- Алгоритм раскройки параболического рефлектора из отдельных сегментов (лепестков).
- Алгоритм раскройки защитного сферического кожуха для параболического рефлектора.
- Реализация алгоритмов раскройки на Python и JavaScript.
- Технология изготовления лепестков из доступных материалов.
- Сборка раскроенных поверхностей.
- Установка параболического рефлектора на чердаке дома (без кожуха).
- Определение оптимального азимута ориентации рефлектора на основе геолокации дома и ближайшей базовой станции сотовой связи.
- Выбор 4G модема и облучателя.

- Принципы строительства мачтовой станции с параболическим рефлектором для улучшенного приема сигнала.
- Принципы создания электронных файлов для лазерной резки лепестков.

Изготовление параболических рефлекторов большого диаметра может быть особенно актуально для отдаленных мест со слабым интернет-сигналом. Приобрести рефлектор диаметром метр и более достаточно сложно, а может быть и вовсе невозможно. Даже если такой рефлектор удастся найти, его транспортировка (если он неразборный) станет крайне затруднительной.

[Репозиторий проекта](#) размещен на платформе [GitHub](#) в открытом доступе, где его можно скачать в полном объеме. Он включает отчеты на русском и английском языках, программные коды на [Python](#) и [JavaScript](#), а также техническую фотогалерею.

## Раскройка лепестков для изготовления параболоида

Геометрические параметры необходимые для расчета лепестков, из которых будет скроен [параболоид вращения](#), показаны на Рис. 1. Простейшее уравнение [параболы](#) в прямоугольной системе координат XZ имеет следующий вид:

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

где  $a$  – положительная или отрицательная константа, определяющая «раскрытие» параболы и направление ее вогнутости. Параболоид получается вращением этой кривой вокруг оси Z. Таким образом, парабола является линией, образующей поверхность. Лепестки раскроя будут образованы в результате рассечения поверхности по линиям параболы, отложенным с одинаковым угловым интервалом вокруг оси вращения.

Каждому радиусу  $r$ , откладываемому от оси вращения, будет соответствовать определенная длина параболы  $l(r)$ , вычисляемая при помощи следующего табличного интеграла:

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

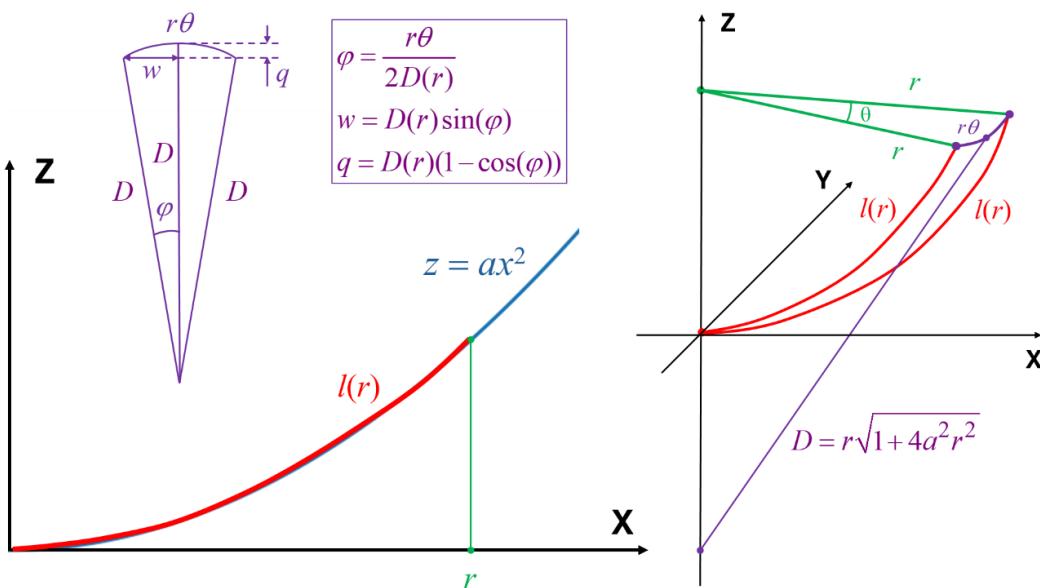
При общем диаметре рефлектора  $D_r = 2R$ , мы получаем для длины  $L$  каждого лепестка:

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

Для определения профиля лепестка нам понадобится определять значение  $r$  по заданному значению  $l$  вдоль параболы, что будет сделано путем численного решения нелинейного уравнения [методом бисекции](#):

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

Алгоритм решения этого уравнения приведен ниже.



**Рис. 1.** Геометрические параметры для расчета лепестков, из которых будет скроен параболоид.

Чтобы понять общие принципы раскройки поверхностей вращения на лепестки, рассмотрим две простые фигуры на Рис. 2 – конус и цилиндр. При разбиении поверхности вращения на  $N$  одинаковых лепестков, угловая ширина каждого из них равна  $\theta = 2\pi/N$  и остается постоянной вдоль лепестка. Для построения профиля лепестка необходимо знать его полную ширину  $2w$  для значения  $s = l - q$  вдоль меридиана лепестка с текущей длинной  $l$ , где  $q$  – высота закругления лепестка, как показано на Рис. 2. В случае конуса ширина лепестка прямо пропорциональна его длине:

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

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

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

Таким образом, профиль лепестка составлен из равностороннего треугольника с закруглением на вершине с радиусом кривизны равным максимальной длине лепестка  $l = L$  вдоль поверхности конуса. В случае цилиндра ширина лепестка постоянна, а радиус кривизны на концах лепестка равен бесконечности. Поэтому его профиль будет представлять собой прямоугольник. Оба эти случая объединяются, если ввести параметр  $D(l)$ , равный длине касательного отрезка, соединяющего вершину лепестка с осью вращения. Радиус кривизны замыкающей дуги лепестка будет равен  $D$ . В случае конуса этот отрезок лежит на наклонной прямой, образующей поверхность, и, следовательно,  $D = l$ . В случае цилиндра этот отрезок также лежит на образующей прямой, однако его длина становится бесконечной, поскольку эта прямая не пересекает ось вращения. Бесконечный радиус кривизны означает прямую линию, в результате чего получается прямоугольный профиль лепестка. Величину  $D$  можно вычислять для конкретных поверхностей вращения. Нас интересует параболоид вращения (см. Рис. 1) и сфера.

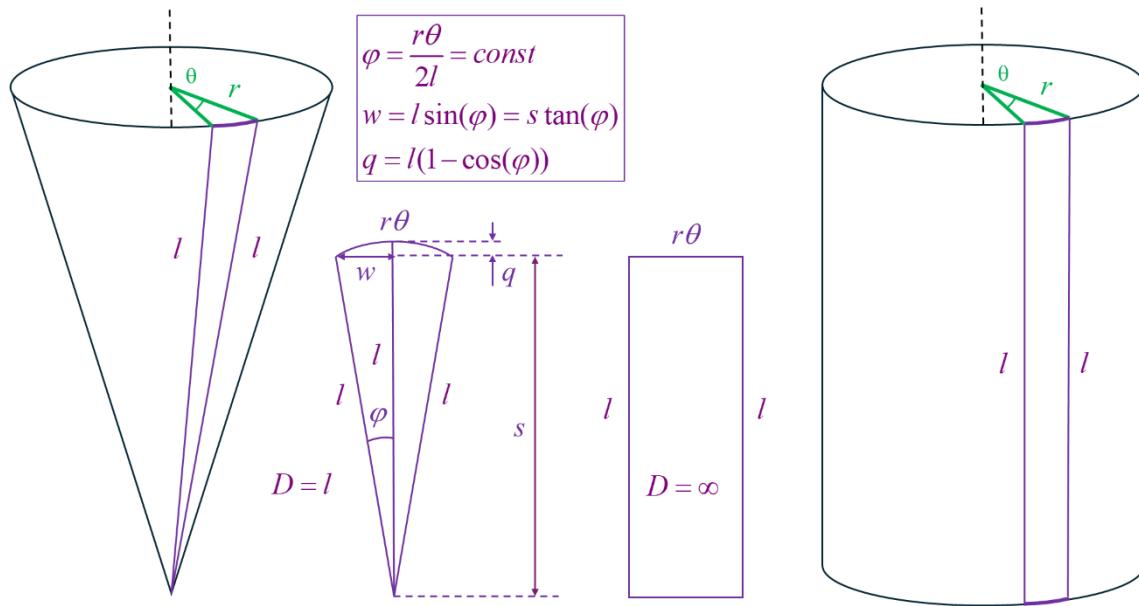


Рис. 2. Раскройка конуса и цилиндра, как простейшие примеры.

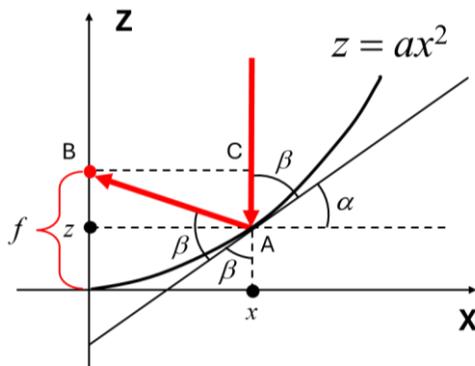
На основе рассмотренных выше примеров раскройки конуса и цилиндра мы можем сформулировать алгоритм построения профиля лепестка параболоида вращения:

1. Вычислить угловую ширину лепестка  $\theta = 2\pi/N$ , где  $N$  – число лепестков.
2. Для заданных  $R$  и  $a$  вычислить общую длину лепестка  $L$  по формуле (3).
3. Вычислить ряд длин лепестков  $l_i = iL/(M - 1)$ ,  $i = \overline{0, M - 1}$ , где  $M$  достаточно большое число разбиений, определяющих точность построения профиля. Нумерация ведется от  $i = 0$ , как это принято в программировании.
4. Вычислить ряд радиусов  $r_i$ , соответствующих  $l_i$ , решая нелинейное уравнение (4) для  $0 < i < M - 1$ . При этом,  $r_0 = 0$  и  $r_{M-1} = R$ .
5. Вычислить ряд касательных отрезков  $D_i = r_i \sqrt{1 + 4a^2 r_i^2}$ ,  $i = \overline{0, M - 1}$ .
6. Вычислить ряд углов раскрыва лепестков  $\varphi_i = \frac{r_i \theta}{2D_i} = \frac{\theta}{2\sqrt{1+4a^2r_i^2}}$ ,  $i = \overline{0, M - 1}$ .
7. Вычислить ряд смещений  $q_i = D_i(1 - \cos(\varphi_i))$ ,  $i = \overline{0, M - 1}$ .
8. Вычислить ряд меридианых координат  $s_i = l_i - q_i$ ,  $i = \overline{0, M - 1}$ .
9. Вычислить ряд ширин  $w_i = D_i \sin(\varphi_i)$ ,  $i = \overline{0, M - 1}$ , откладываемых от соответствующих меридианых координат  $s_i$  в обе стороны.
10. Для замыкания лепестка соединить концы конечных ширин  $w_{M-1}$  с вершиной лепестка  $s_M = L$  (дополнительная меридианская координата) дугой с радиусом кривизны  $D_{M-1} = R\sqrt{1 + 4a^2R^2}$ .

В качестве основных параметров параболоида вращения мы будем использовать максимальный радиус рефлектора  $R$  и его фокусное расстояние  $f$ . В качестве упражнения докажем хорошо известную формулу для вычисления параметра параболы  $a$  через  $f$ :

$$a = 1/(4f) \quad (8)$$

Для этого обратимся к Рис. 3, наглядно показывающим свойство параболоида отражать лучи, падающие параллельно его оси, в одну точку, называемую фокусом. Путь одного такого луча показан красными линиями на рисунке.



**Рис. 3.** Геометрические параметры, используемые для вычисления фокусного расстояния  $f$ .

Для касательного угла  $\alpha$  и угла падения луча  $\beta$  получаем:

$$\tan(\alpha) = z'(x) = 2ax \quad (9)$$

$$\beta = \frac{\pi}{2} - \alpha \quad (10)$$

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

Далее, для прямоугольного треугольника ABC имеем следующее отношение (угол падения равен углу отражения относительно касательной линии в точке падения луча на поверхности):

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

Отношение (12) остается справедливым для любого положения точки  $x$ , в том числе, когда точка  $z$  оказывается выше фокуса. Треугольник ABC был использован лишь для вывода отношения.

Используя уравнения (9), (11), и (12), получаем окончательно:

$$\frac{x}{f-z} = \frac{2 \tan(\alpha)}{1 - \tan^2(\alpha)} \Rightarrow \frac{x}{f-ax^2} = \frac{4ax}{1-4a^2x^2} \Rightarrow \frac{1}{f-ax^2} = \frac{1}{\frac{1}{4a}-ax^2} \Rightarrow f = \frac{1}{4a} \quad (13)$$

Таким образом, мы доказали, что фокус действительно существует, поскольку он не зависит от точки падения параллельного луча на параболоид.

Профиль лепестка будет образован таблицей значений  $(s_i, w_i)$  и замыкающей дугой – шаги алгоритма 8–10. Ниже мы приводим программный код на Python для расчета профиля. Он может быть скачан из [репозитария проекта](#) (файл Parabolic\_reflector.py). Для работы программы понадобятся две Python библиотеки [csv](#) и [numpy](#). Для отладки и запуска программы можно использовать любую оболочку [IDE](#). Мы предпочитаем работать в бесплатной версии [PyCharm Community Edition](#), которая позволяет легко подгружать необходимые библиотеки. Ниже в этом отчете мы рассмотрим программные коды на [JavaScript](#), которые избавят от необходимости использовать Python, если вы не знакомы с этим языком программирования.

```
1. #
2. # Algorithm for cutting the petals for a parabolic reflector
3. #
4. # Dmitriy Makhnovskiy, August 2024
5. #
6.
7. import csv
8. import numpy as np
9.
10. pi = np.pi # pi-constant 3.1415....
11. c = 2.99792458e8 # speed of light in m/s
12.
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 in GHz
23. f_max = 2.6 # maximum in GHz
24. k = 0.6 # reflector efficiency (0-1)
25.
26. # Calculated parameters
27. f_min = f_min * 1.0e9 # frequency in Hz
28. f_max = f_max * 1.0e9 # frequency in Hz
29.
30. lambda_max = c / f_min # maximal wavelength
31. lambda_min = c / f_max # minimal wavelength
32. if units == 'mm':
33.     radius = R / 1000.0
34. elif units == 'cm':
35.     radius = R / 100.0
36. else:
37.     radius = R
38.
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)
42.
43. a = 1.0 / (4.0 * f) # parabola coefficient, z(x) = a*x^2
44. theta = 2.0 * pi / N # angular width of the petal
45. value = np.sqrt(1.0 + 4.0 * a**2 * R**2)
46. L = R * value / 2.0 + np.log(2.0 * a * R + value) * f # petal length
47.
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
67.
68. l = [0.0] * M # points along the petal
69. l[M - 1] = L
70.
71. r = [0.0] * M # array of r corresponding to the array of l
72. r[M - 1] = R
73.
74. D = [0.0] * M # array of the tangent segments
75. D[M - 1] = R * np.sqrt(1.0 + 4.0 * a**2 * R**2)
76.
77. eff = [0.0] * M # array of the opening angles
78. eff[0] = theta / 2.0
79. eff[M - 1] = theta / (2.0 * np.sqrt(1.0 + 4.0 * a**2 * R**2))
80.
81. q = [0.0] * M # meridian displacements with respect to the l-points
82. q[M - 1] = D[M - 1] * (1.0 - np.cos(eff[M - 1]))
83.
84. s = [0.0] * M # array of the meridian coordinates
85. s[M - 1] = l[M - 1] - q[M - 1]
86.
87. w = [0.0] * M # array of the widths corresponding to the array of s
88. w[M - 1] = D[M - 1] * np.sin(eff[M - 1])
89.
90. for i in range(1, M-1):
91.     length = i * L / (M - 1)
92.     l[i] = length
93.     r[i] = rl(length)
94.     D[i] = r[i] * np.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2)
95.     eff[i] = theta / (2.0 * np.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2))
96.     q[i] = D[i] * (1.0 - np.cos(eff[i]))
97.     s[i] = l[i] - q[i]
98.     w[i] = D[i] * np.sin(eff[i])
99.
100. # Saving the profile data to a CSV file
101. data = np.column_stack((l, r, D, eff, q, s, w))
102. header = ['l', 'r', 'D', 'eff', 'q', 's', 'w']
103.
104. with open('Parabolic_profile_data.csv', 'w', newline='') as csv_file:
105.     writer = csv.writer(csv_file)
106.     writer.writerow(header)
107.     writer.writerows(data)
108.
109. # Saving the design parameters to a txt file
110. Dish_radius = 'Dish radius = ' + str(R) + ' ' + units + '\n'
111. Dish_height = 'Dish height = ' + str(a * R**2) + ' ' + units + '\n'
112. Focus_length = 'Focus length of the reflector= ' + str(f) + ' ' + units + '\n'
113. Maximum_length = 'Petal length = ' + str(L) + ' ' + units + '\n'
114. N_number = 'Number of petals = ' + str(N) + '\n'
115. Petal_radius = 'Petal radius of curvature = ' + str(D[M - 1]) + ' ' + units + '\n'
116. Frequency_min = 'Minimum frequency = ' + str(f_min / 1.0e9) + ' GHz' + '\n'
117. Frequency_max = 'Maximum frequency = ' + str(f_max / 1.0e9) + ' GHz' + '\n'
118. Wavelength_min = 'Minimum wavelength = ' + str(lambda_min) + ' m' + '\n'
119. Wavelength_max = 'Maximum wavelength = ' + str(lambda_max) + ' m' + '\n'
120. Reflector_eff = 'Reflector efficiency = ' + str(k) + '\n'
121. Antenna_Gain_min = 'Minimum antenna gain = ' + str(Gain_min) + ' dB' + '\n'
122. Antenna_Gain_max = 'Maximum antenna gain = ' + str(Gain_max) + ' dB' + '\n'
123.
124. design_parameters = open('Design_parameters.txt', 'w')
125. design_parameters.write(Dish_radius)
126. design_parameters.write(Dish_height)
127. design_parameters.write(Focus_length)

```

```

128. design_parameters.write(Maximum_length)
129. design_parameters.write(N_number)
130. design_parameters.write(Petal_radius)
131. 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()

```

Программа представляет собой консольное приложение, поэтому не имеет пользовательского интерфейса. Все параметры (Design parameters) необходимо ввести непосредственно в текст программы, как объяснено ниже:

```

13. # Design parameters
14. units = 'mm' # Единицы измерения длин: мм, см, или м. Это текстовая переменная, поэтому она вводится в кавычках.
15. # Для R и f используйте одинаковые единицы: мм, см, или м.
16. R = 500.0 # Радиус рефлектора (здесь использованы мм).
17. f = 500.0 # Фокусная длина (здесь использованы мм).
18. N = 10 # Число лепестков для раскройки. Выбирайте это значение, исходя из свойств материала, из которого будут изготавливаться лепестки.
19. M = 50 # Число точек вдоль лепестка, используемых для построения его профиля. Чем больше точек, тем выше точность профиля. Однако при ручном построении, число точек придется взять не очень большим.
20. eps = 1.0e-10 # Точность вычисления корня нелинейного уравнения (4).
21. # Минимальная и максимальная частоты связи, которые используются для интернета (2G, 3G, 4G, 5G). Например, для 4G LTE диапазон частот охватывает от 900 МГц до 2.6 ГГц. Хотя эти параметры никак не используются для расчета профиля лепестка, они позволяют оценить коэффициент усиления рефлектора.
22. f_min = 0.9 # Минимальная частота в ГГц
23. f_max = 2.6 # Максимальная частота в ГГц
24. k = 0.6 # Коэффициент эффективности рефлектора в диапазоне (0-1). Выбор этого значения обсуждается ниже.

```

Алгоритм метода бисекции (или «метод деления пополам») для решения нелинейного уравнения (4) реализован виде функции rl(length). Эта функция будет вызываться каждый раз, когда надо найти текущий радиус  $r$  по текущей меридианной координате  $l$ . Точность вычисления корня контролируется абсолютной величиной разности  $\text{eps} = 10^{-10}$  в (4). Максимальное число итераций при поиске корня уравнения ограничено 100, что исключает возможную «осцилляцию» около корня, если заданная точность вычислений оказалась слишком высокой.

```

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

```

В результате исполнения программы создается табличный файл Parabolic\_profile\_data.csv, см. Рис. 4, столбцы которого содержат все дискретные параметры, вычисляемые в алгоритме («eff» означает  $\varphi$ ).

Для построения профиля лепестка ( $s_i, w_i$ ) понадобятся только два последних столбца s и w. Радиус кривизны замыкающей дуги равен последнему значению в столбце D.

1	l	r	D	eff	q	s	w
2		0	0	0	0.314159	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	106.1459	105.0470	106.5400	0.312411	5.157071	100.0887	32.74572

**Рис. 4.** Структура табличного файла Parabolic\_profile\_data.csv, создаваемого программой.

Также программа сохраняет все дизайн параметры в текстовом файле Design\_parameters.txt. Пример содержимого этого файла показан ниже для рефлектора радиусом 500 мм и фокусным расстоянием 500 мм:

Dish radius = 500.0 mm (радиус рефлектора)

Dish height = 125.0 mm (глубина рефлектора, отсчитываемая от полюса)

Focus length of the reflector = 500.0 mm (фокусная длина)

Petal length = 520.1144097172755 mm (длина лепестка от истока до макушки замыкающей арки)

Number of petals = 10 (число лепестков)

Petal radius of curvature = 559.0169943749474 mm (радиус кривизны замыкающей арки лепестка)

Minimum frequency = 0.9 GHz (минимальная частота)

Maximum frequency = 2.6 GHz (максимальная частота)

Minimum wavelength = 0.11530479153846154 m (минимальная длина волны)

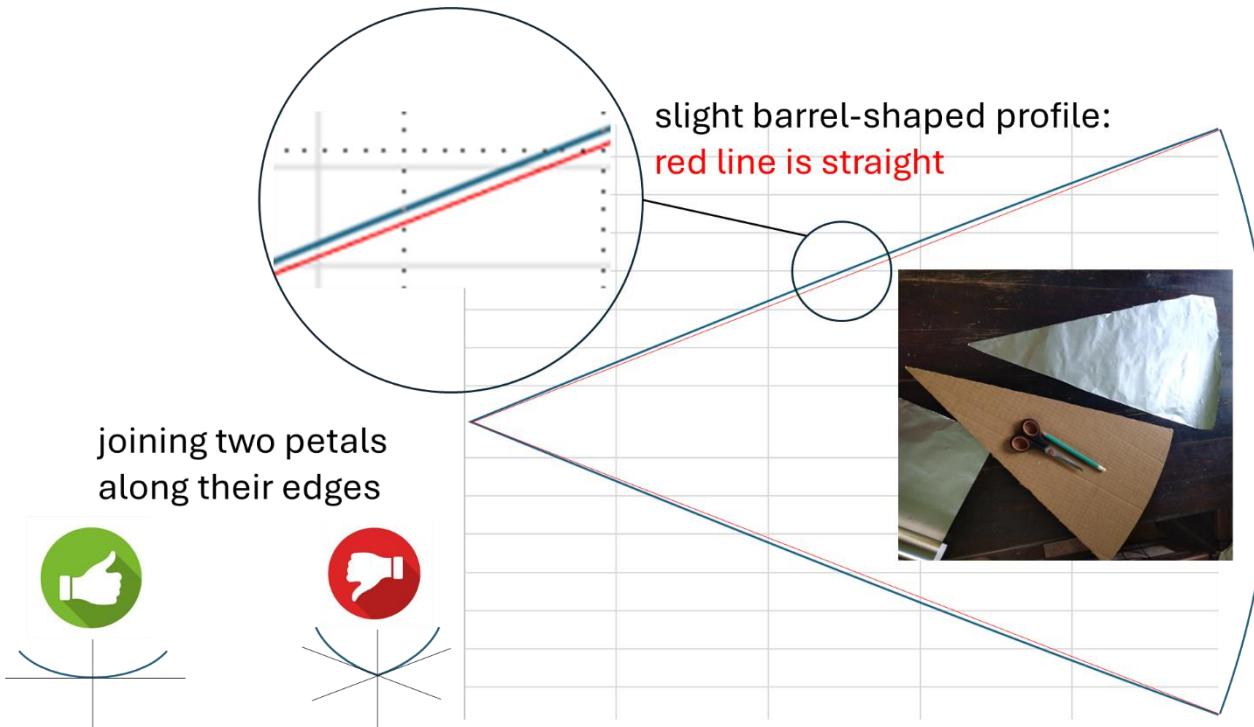
Maximum wavelength = 0.3331027311111111 m (максимальная длина волны)

Reflector efficiency = 0.6 (эффективность рефлектора)

Minimum antenna gain = 17.27294608794706 dB (минимальное усиление рефлектора)

Maximum antenna gain = 26.487562858576922 dB (максимальное усиление рефлектора)

Профиль лепестка, рассчитанный для этих параметров, показан на Рис. 5. Хотя он и очень близок к треугольному (красные прямые линии), но имеет небольшую выпуклость наружу («бочкообразный»).



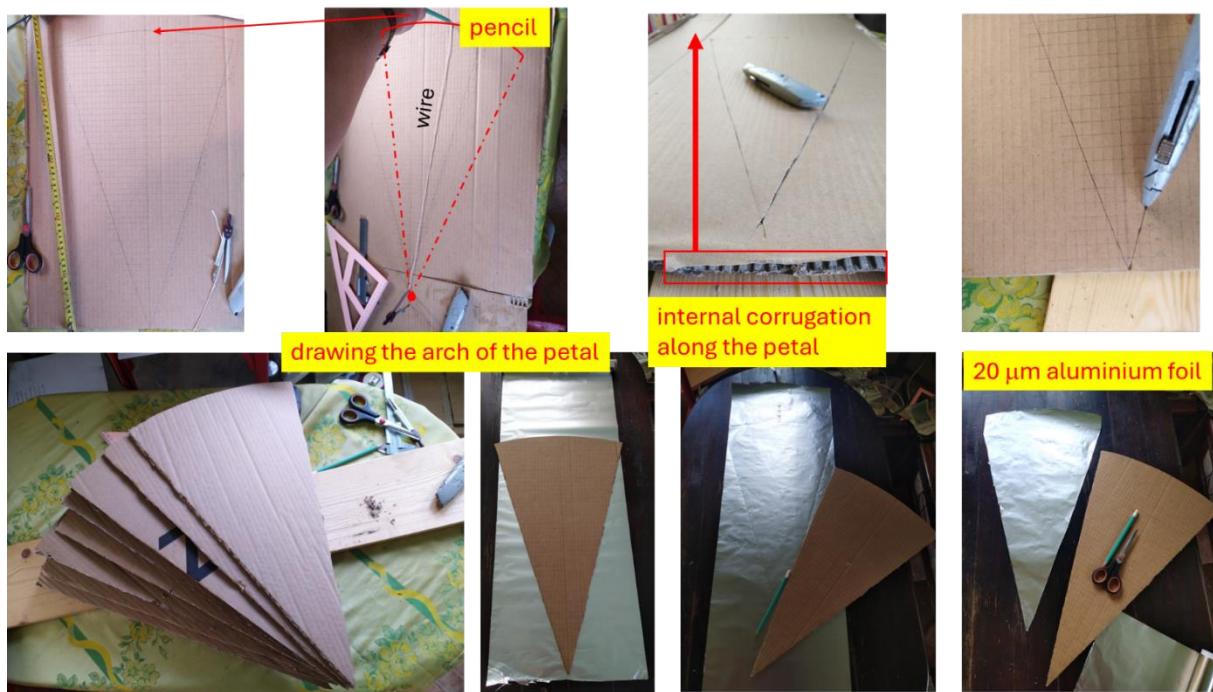
**Рис. 5.** Профиль лепестка для параболоида радиусом 500 мм и фокусным расстоянием 500 мм.

Точность воспроизведения параболической поверхности является одним из ключевых факторов, влияющих на коэффициент эффективности рефлектора  $k$  (reflector efficiency), хотя и не единственным. Поэтому важно вырезать лепестки с максимальной аккуратностью, что представляет значительные трудности при ручной работе по шаблону. Сочленение лепестков должно происходить строго по швам, так, чтобы в каждой точке вдоль шва оба лепестка имели общую касательную плоскость, как показано на Рис. 5. Наш первый прототип рефлектора был собран из лепестков, вырезанных из упаковочного картона. Очевидно, что нам не удалось достичь высокой точности вырезки. Более того, при склеивании мы использовали липкую ленту, которая не могла идеально удерживать лепестки в одной касательной плоскости по шву сочленения, т.е. присутствовал небольшой излом. Хотя итоговая форма оказалась вполне приемлемой, она все еще далека от идеала. Технологические аспекты изготовления рефлектора обсуждаются в следующем разделе.

## Технология изготовления параболического рефлектора из лепестков

Наш первый прототип рефлектора мы изготовили из упаковочного картона. Возможно, это может показаться забавным, но напомним, что во время Второй Мировой Войны из фанеры делали истребители — главное, чтобы летало и стреляло! В нашем случае результат оказался схожим: форма рефлектора получилась вполне функциональной. Основные проблемы, как уже упоминалось в предыдущем разделе, заключались в точности вырезания самого шаблона и лепестков по нему, а также в касательном склеивании по швам. Подробная фотогалерея о нашем опыте размещена в [репозитарии проекта](#) в папке Design gallery.

Этапы раскройки лепестков показаны на Рис. 6. Шаблон был вычерчен на основе рассчитанной таблицы значений  $(s_i, w_i)$ , где сначала вдоль прямой откладывались точки  $s_i$ , а затем для каждой из них перпендикулярно прямой с обеих сторон откладывались соответствующие ширины  $w_i$ . До тех пор, пока позволяла длина  $w_i$ , использовался более точный (доли миллиметра) штангенциркуль. Для больших ширин пришлось использовать линейку, что менее точно. После этого концы перпендикулярных отрезков соединялись плавными линиями, интерполирующими профиль лепестка. Замыкающая арка лепестка вычерчивалась при помощи проволоки-радиуса (см. Рис. 6). Остальные лепестки вырезались ножом по карандашному контуру, нанесенному с использованием шаблона. Что важно для последующей сборки, профиль лепестка должен быть ориентирован вдоль внутренних гофр картона (см. Рис. 6). Это обеспечит упругость лепестков, позволяя им лучше следовать параболической линии и сохранять заданную форму. Тот же шаблон использовался для разметки лепестков из кухонной алюминиевой фольги толщиной 20 микрон, которыми выстилалась отражающая поверхность параболоида.



**Рис. 6.** Раскройка картонных лепестков, из которых был собран рефлектор.

Этапы склеивания параболоида представлены на Рис. 7. Для соединения лепестков использовалась липкая лента, благодаря чему оболочка стала самонесущей. Однако, как отмечалось ранее, данный метод не обеспечивает идеального касательного сопряжения лепестков вдоль швов (см. Рис. 5), так как лента недостаточно жесткая. Для защиты от влаги склеенная поверхность была пропитана акриловым лаком с обеих сторон. После высыхания для увеличения прочности внутренняя поверхность была дополнительно оклеена газетной бумагой, а внешняя — стеклообоями, вырезанными по форме лепестков. На заключительном этапе к поверхности был прикреплен пластиковый трубчатый каркас для усиления жесткости конструкции и обеспечения возможности её подвески. Для отражения электромагнитных волн внутренняя поверхность параболоида была обклеена лепестками из алюминиевой фольги толщиной 20 микрон. Для всех клеевых работ применялся ПВА.



**Рис. 7.** Этапы склеивания параболоида из лепестков, дополнительная обработка поверхностей, и выстилание внутренней поверхности алюминиевой фольгой.

Процесс сборки двухярусного трубчатого каркаса показан на Рис. 8. Мы использовали короткие отрезки пластиковых трубок с внешним диаметром около 10 мм. Для их удлинения трубы соединялись между собой с помощью внутренних муфт, изготовленных из тех же трубок с продольным разрезом. Место стыка обматывалось липкой лентой для укрепления соединения. Круглый трубчатый обруч диаметром 1 м был подшит к кромке параболоида с помощью тонкой проволоки в оплетке. Радиальные трубы первого и второго ярусов крепились к внешнему обручу с

помощью концевых хомутов (половина трубы), а в месте их пересечения стягивались сквозным болтом с шайбами, проходящим через полюс параболоида. Дополнительно трубы первого яруса были прикреплены проволокой к поверхности в нескольких точках для усиления конструкции. После завершения сборки внешняя поверхность рефлектора была окрашена белой масляной краской.

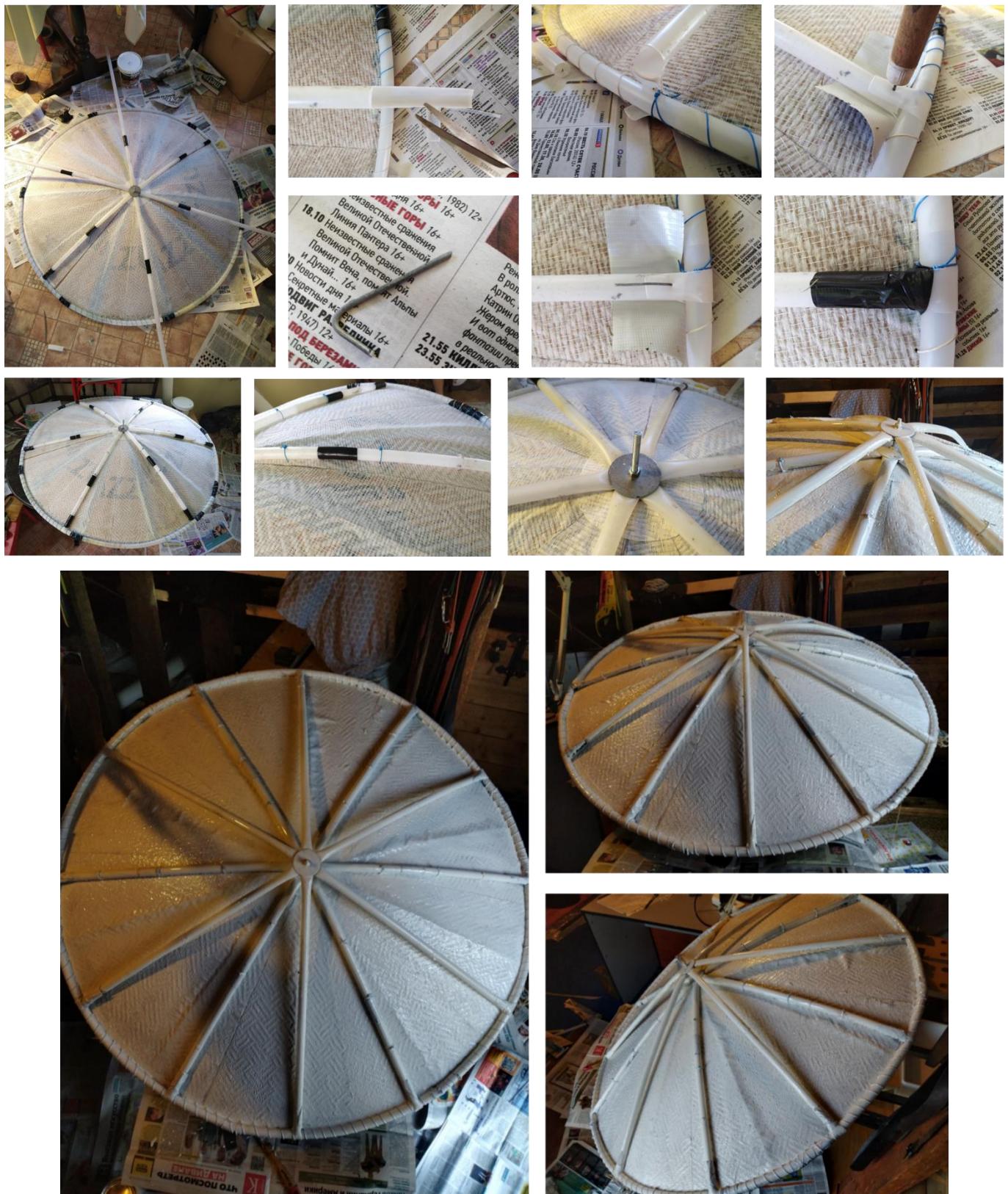
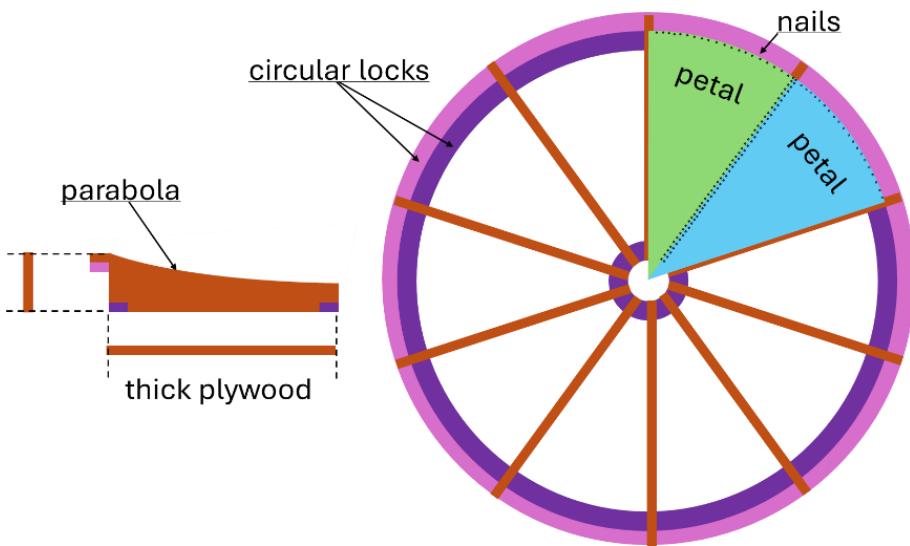


Рис. 8. Трубчатый каркас для усиления рефлектора и его подвески.

Опробованный нами метод изготовления рефлектора не следует рассматривать как некий темплет для последующих моделей. Скорее, он помог выявить проблемы и предложить пути их решения. В первую очередь необходимо провести эксперименты с материалами для лепестков. Это могут быть тонкая фанера, шпон, акрил, поликарбонат, другие виды пластика или даже тонкий стальной лист. Для обеспечения максимальной точности и воспроизводимости, резка лепестков и других деталей каркаса должна осуществляться [лазером](#), что сегодня доступно даже через небольшие компании при любом объеме заказа. Особенно привлекательна идея создания единственной высокоточной формы для последующего изготовления композитных оттисков. Технология производства композитных оболочек с применением [вакуумной инфузии](#) заключается в укладке слоев армирующих материалов, таких как стекловолокно или углепластик, на форму, после чего они пропитываются полимерной матрицей, чаще всего эпоксидной смолой. Затем вся конструкция помещается в вакуумный мешок, откуда удаляется воздух, что обеспечивает равномерное распределение смолы и исключает образование воздушных пузырей. Вакуум способствует глубокой инфузии смолы в армирующий материал, что улучшает адгезию и прочностные характеристики композита. Финальной стадией является термообработка, которая завершает процесс полимеризации. Этот метод позволяет создавать легкие и прочные конструкции с высокой точностью и однородностью.

Из вышеизложенного следует, что в любом случае понадобится точная форма. Одной из ключевых проблем является касательное сочленение лепестков (см. Рис. 5), которое должно быть реализовано по жестким ребрам каркаса, притягивающим швы лепестков к единой касательной плоскости в каждой точке вдоль параболы. На Рис. 9 предложена концепция рамы из фанеры, способной обеспечить высокую точность воспроизведения поверхности параболоида. Все детали, включая каркас из толстых листов 10–15 мм и тонкие лепестки 2–3 мм (фанера или шпон), должны быть вырезаны лазером. Для крепления лепестков можно использовать быстроотвердевающий клей в сочетании с короткими гвоздиками. В процессе монтажа лепестки следует поджимать в направлении выпуклости поверхности, чтобы кромки располагались точно на середине ребер и по периметру внешнего кольца. Это обеспечит их касательное сочленение. После изготовления форму необходимо зашкурить по швам и покрыть водонепроницаемым лаком, что позволит многократно использовать ее для создания композитных оболочек методом вакуумной инфузии. Готовую композитную оболочку скорее всего придется дополнительно усилить легким каркасом, аналогичным использованному нами, чтобы предотвратить деформацию во время монтажа. Эти идеи можно развивать и дальше, продолжая эксперименты с материалами, дизайном, и технологиями изготовления.



**Рис. 9.** Концепция фанерной рамы, обеспечивающая высокую точность воспроизведения поверхности параболоида. Все детали вырезаются лазером.

Конструкцию на Рис. 9, если она предназначена для изготовления композитных оттисков, следует рассматривать как «негативную форму», где оттиск располагается внутри вогнутости, а рама находится снаружи. В случае «позитивной формы» оттиск окажется снаружи вогнутости, а рама внутри нее. Принцип раскройки лепестков останется неизменным, однако прижимание их кромок к ребрам рамы «позитивной формы» может потребовать более кропотливой работы.

Позитивные формы могут оказаться более удобными для композитных работ. Мы хотели бы ознакомить вас с одним [методом создания стационарной позитивной формы](#) без использования раскройки на лепестки, который мы нашли на YouTube-канале [вьетнамского умельца](#) (хотя наверняка это уже не индивидуал, а целая команда). Сначала выпиливают внешний профиль изделия (из фанеры или пластика) и закрепляют его на вертикальной стойке с возможностью вращения, как показано на Рис. 10. Вокруг этой стойки насыпается слой сырого песка, который выравнивается вращающимся профилем для придания предварительной формы. Затем по уже спрофилированной поверхности разливается жидкий раствор цемента, который также выравнивается путем вращения профиля. После затвердевания раствора образуется прочная корка с желаемой формой. На эту корку можно укладывать стеклоткань и пропитывать ее эпоксидной смолой. Этот метод можно попробовать в вашей практике. Однако, несмотря на кажущуюся простоту, для формирования такой профильной корки потребуется значительная сноровка. Кроме того, есть очевидные недостатки: невозможность транспортировки формы, трудности со хранением и отсутствие возможности размещения формы в термической камере для вулканизации эпоксидной смолы, что ограничивает возможность улучшения процесса.

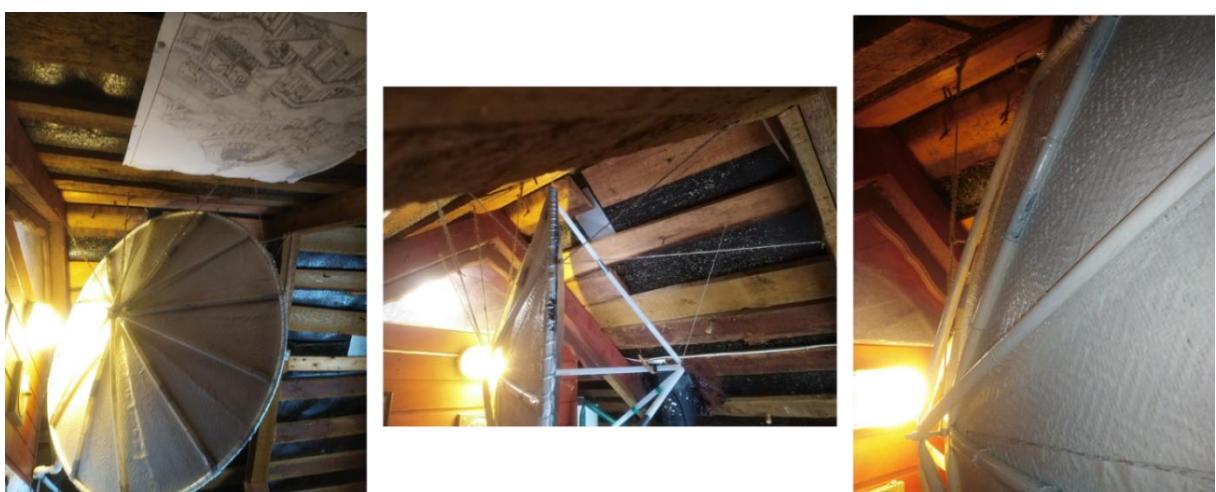


**Рис. 10.** Создание стационарной позитивной формы из песка и цемента для оттиска композитов.

Композитные технологии сегодня стали настолько доступными, что их можно применять даже в условиях небольшой мастерской. Использование таких методов, как вакуумная инфузия или вакуумное формование, не требует сложного оборудования и позволяет создавать прочные и легкие конструкции. Благодаря доступности материалов, таких как стекловолокно, углепластик и эпоксидные смолы, а также возможности лазерной резки для точного изготовления форм, изготовление композитных деталей стало возможным практически для любого энтузиаста. Это открывает широкие перспективы для индивидуальных проектов и экспериментов с новыми технологиями.

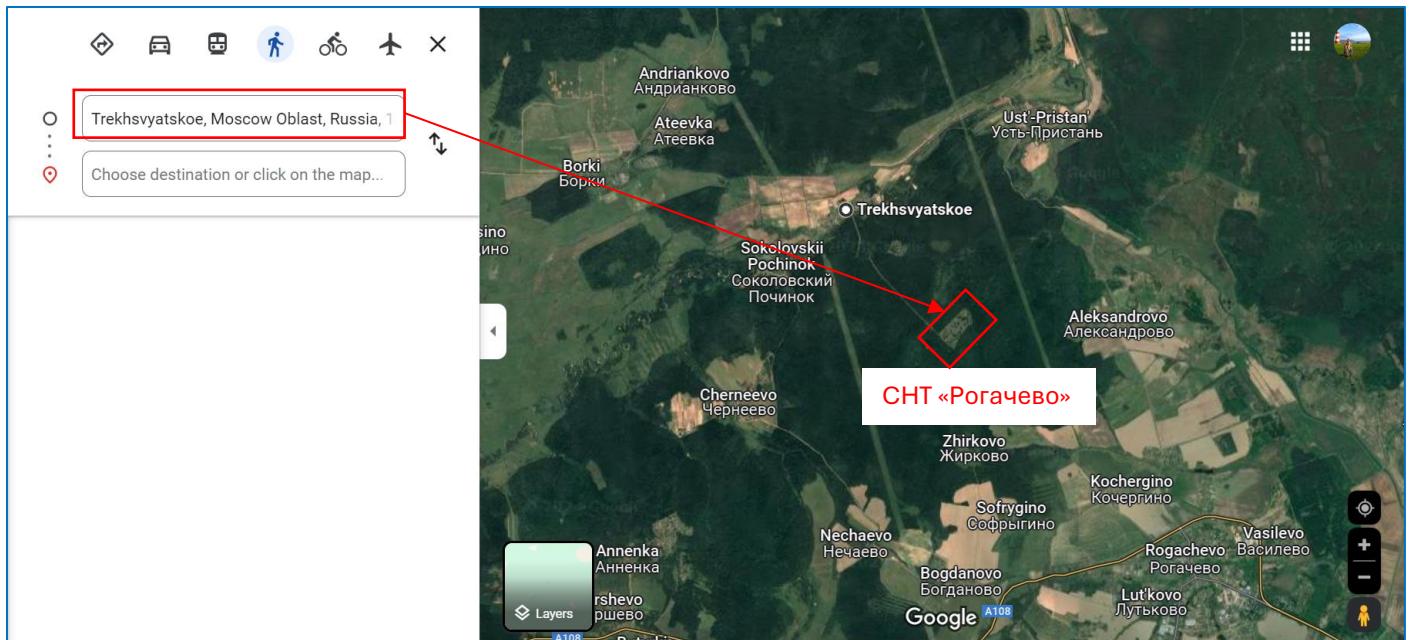
### Монтаж рефлектора и его ориентация на вышку связи

Изготовленный нами рефлектор оказался настолько легким, что его можно было подвесить на тонких веревках на чердаке, зафиксировав направление на базовую станцию, как показано на Рис. 11. Поэтому мы отказались от использования жестких кронштейнов на этом этапе проекта.



**Рис. 11.** Подвешивание рефлектора на чердаке дома.

Ориентация рефлектора проводится в несколько этапов, как объяснено ниже. Сначала, используя [Google maps](#), найдите садоводческое товарищество, которое будет видно со спутника.



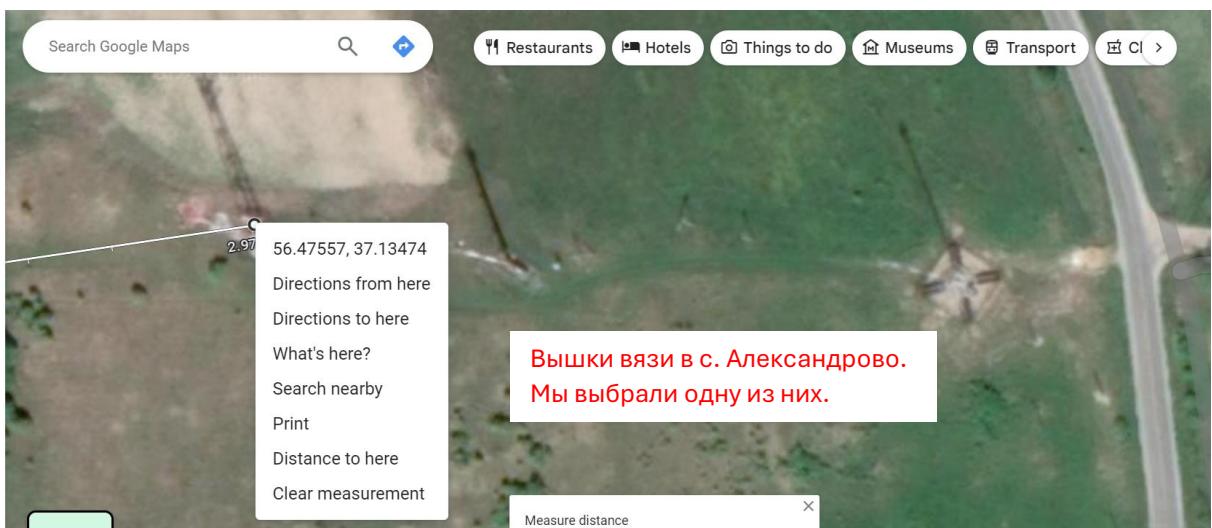
Увеличьте масштаб и вы будете в состоянии увидеть ваш участок и даже дом. Используя правую клавишу мышки, откройте меню, показанное ниже, и выберите «measure distance» (измерить дистанцию). Появится референс точка, которую можно передвинуть мышкой на ваше строение, где установлена антенна.



Наведите курсор мышки на эту точку и опять кликните на правую клавишу мышки. В открывшемся меню вы увидите геодезические координаты вашего дома.



Не трогая первую референс точку, уменьшите масштаб (колесиком на мышке), перейдите на карте в район, где установлена вышка связи (двигая карту мышкой), опять увеличьте масштаб, и найдите вышку. Затем, используя правую клавишу мышки вызовите меню и выберите «distance to here» (дистанция до сюда). Появится вторая референс точка, которую надо передвинуть мышкой на вышку связи. Далее указывая курсором мышки на эту точку и кликая правую клавишу мышки, вы получите геодезические координаты базовой станции.



Уменьшите масштаб (колесиком мышки), чтобы увидеть полное направление и дистанцию от вашего дома до вышки.



И так, мы определили:

Дистанция от дома 62 до вышки связи – 2 км 970 м.

Геодезические координаты дома 62 – (56.47166, 37.08682)

Геодезические координаты вышки в с. Александрово – (56.47557, 37.13474)

Теперь необходимо перевести дробные (десятичные) значения в градусы, минуты, и секунды, что можно сделать на [вебсайте](#), как показано ниже.

Перевод дробного значения градусов в градусы минуты и секунды

Градусы  
56.47166

Показывать доли секунд

Градусы, заданные десятичной дробью

Градусы, минуты и секунды  
56°28'18"

РАССЧИТАТЬ

В результате мы получим:

Геодезические координаты дома 62 – 56°28'18" С. Ш. и 37°5'13" В. Д. (Северной Широты и Восточной Долготы)

Геодезические координаты вышки в с. Александрово – 56°28'32" С. Ш. и 37°8'5" В. Д.

Используя эти координаты и [вебсайт](#), мы можем вычислить азимут на вышку связи, как показано внизу.

Вычисление постоянного азимута и длины линии румба

Начальная точка, широта  
56° 28' 18"  с.ш.  ю.ш.

Начальная точка, долгота  
37° 5' 13"  в.д.  з.д.

Конечная точка, широта  
56° 28' 32"  с.ш.  ю.ш.

Конечная точка, долгота  
37° 8' 5"  в.д.  з.д.

Референц-эллипсоид  
 WGS-84  SK-42  Шар

Точность вычисления  
Знаков после запятой: 2

РАССЧИТАТЬ

Азимут 81.63	Расстояние в километрах 2.98	Расстояние в морских милях 1.61
-----------------	---------------------------------	------------------------------------

Мы получили, что географический азимут от дома 62 на вышку в с. Александрово составляет  $81.63^{\circ}$  на Восток. Однако если мы хотим измерять этот азимут, используя компас, то необходимо учесть магнитное склонение (magnetic declination), поскольку Магнитный Север не совпадает с Географическим Севером. Магнитное склонение можно вычислить на сайте [National Centers for Environmental Information](#), введя ваши координаты в десятичной форме, как показано ниже.

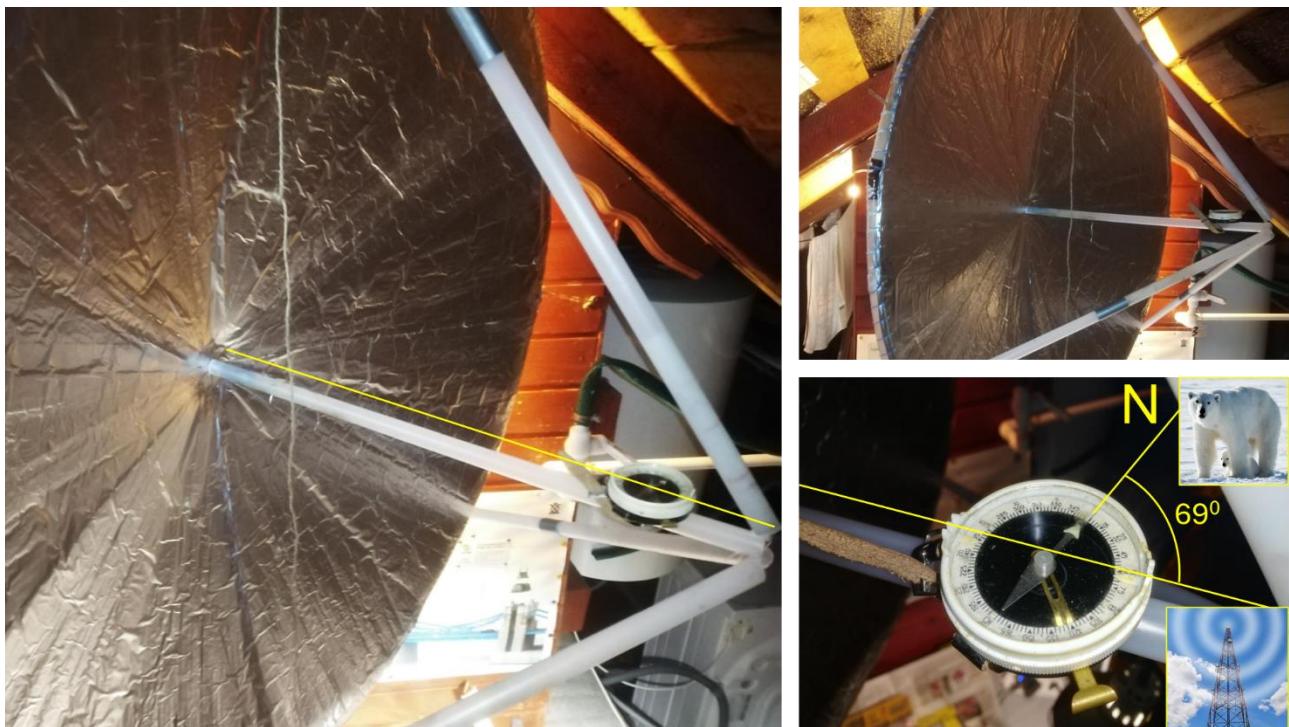
Для нашего СНТ «Рогачево» магнитное склонение составляет  $\sim 12.35^{\circ}$  на Восток, причем это значение «плывет» на  $0.14^{\circ}$  на Восток каждый год. Поэтому, читая этот отчет, все же уточните текущее значение на вебсайте, хотя такая высокая точность нам не понадобится. Таким образом, магнитный азимут от дома 62 на вышку в с. Александрово составляет  $81.63 - 12.35 = 69.28^{\circ}$ . Именно он и будет использован для наведения рефлектора. Заметим, что это направление будет варьироваться от дома к дому, поскольку площадь СНТ не так уж мала, по сравнению с дистанцией до вышки.

На смартфоне магнитное склонение можно определить автоматически, поскольку он засечет вашу текущую геолокацию. Просто найдите в поисковике браузера ссылку на [NOAA Magnetic Field Calculator](#). Ниже показан скриншот со смартфона, который был сделан на углу СНТ, где еще брался интернет. Из него видно, что Магнитный Север **M** смещен относительно Геодезического Севера **N** на  $+12.3^{\circ}$ , где положительный знак означает на Восток.



Антенны направленного типа, особенно узконаправленные, такие как параболический рефлектор, требуют точной ориентации на базовую станцию, а не на локальный интерференционный максимум, который может изменяться с течением времени. Поэтому такие антенны не следует вращать вокруг оси, а лишь корректировать их направление в пределах малых отклонений от рассчитанного азимута на базовую станцию. Чем выше расположена антenna, тем точнее можно осуществить наведение. Для измерения уровня сигнала с модема можно использовать смартфон, однако его показания могут иметь задержку, что делает этот метод менее точным. Поэтому корректировка направления должна выполняться очень медленно. Идеальным вариантом было бы использование электромеханического привода для управления вращением антенны, но это доступно не каждому. В большинстве случаев приходится довольствоваться ручной регулировкой.

Наша схема ручного наведения рефлектора на вышку связи по магнитному азимуту показана на Рис. 12. Рефлектор был снабжен фронтальными стойками из пластиковых трубок, которые явно обозначают его фокус, и к которым крепится модем. На центральной горизонтальной трубке был установлен компас с азимутальными маркерами, показывающими направление  $69^0$  на Восток. Затем, вращая подвешенный рефлектор, мы добились совмещения магнитной стрелки с Севером. Это направление было зафиксировано дополнительными растяжками.



**Рис. 12.** Ручное наведение рефлектора на вышку связи по магнитному азимуту.

## Оснащения рефлектора 4G модемом и облучателем

Параболический рефлектор, используемый в телекоммуникационных приложениях, выполняет функции как приема, так и передачи сигналов. Что касается самой поверхности отражения, то она эффективно работает с радиоволнами любой поляризации. Однако рефлектор очень чувствителен к ориентации относительно базовой станции, поскольку обладает узкой диаграммой направленности. Рефлектор может быть оснащен различными типами облучателей, которые доступны на рынке, что предоставляет широкие возможности для экспериментов. При наличии знаний можно попробовать разработать собственный облучатель.

В режиме передачи облучатель принимает сигнал от модема через кабель и формирует радиальный электромагнитный волновой фронт с определенной поляризацией. Этот фронт, падая на параболическую поверхность рефлектора, переотражается, создавая направленный луч, параллельный оси рефлектора, который направляется в сторону базовой станции. В режиме приема облучатель улавливает электромагнитное излучение, сфокусированное рефлектором, и преобразует его в сигнал, который затем передается модему по кабелю.

За счет фокусировки волн с большой апертурой рефлектора на малую апертуру облучателя в его фокусе происходит усиление мощности сигнала. Коэффициент усиления мощности рефлектора (измеряемый в [декибелах](#), dB или dB) вычисляется по [формуле](#):

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

где  $R$  радиус рефлектора,  $\lambda$  длина электромагнитной волны, и  $k$  коэффициент эффективности рефлектора. Эта формула уже была использована в программе расчета:

```
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)
```

Обратите внимание, что идеальная фокусировка в одну точку невозможна в контексте волновой оптики, где длина волны не является бесконечно малой, в отличие от геометрической оптики, использующей идеальные световые лучи. В волновой оптике фокус представляет собой объем с поперечным размером порядка длины волны  $\lambda$ . Чем меньше длина волны (и, соответственно, выше частота), тем меньше этот объем фокусировки, что приводит к увеличению коэффициента усиления. Для стандарта 4G зарезервирован широкий диапазон частот, от 900 МГц до 2.6 ГГц, что соответствует длинам волн от примерно 0.33 м до 0.12 м. Соответственно, усиление сигнала может варьироваться в широких пределах в зависимости от несущей частоты.

Коэффициент  $k$ , указанный в уравнении (14), наряду с другими факторами зависит от точности воспроизведения формы параболоида и гладкости его поверхности после фольгирования. В нашем картонном рефлекторе эти параметры имели посредственное качество. Мы уже наметили пути решения этих проблем. Дизайн облучателя, который представляет из себя пассивную антенну, также будет влиять на величину  $k$ :

- Облучатель должен быть спроектирован так, чтобы его форма и размер соответствовали геометрии рефлектора. Несоответствие может привести к неправильному распределению излучения и, как следствие, к снижению эффективности. Например, если облучатель слишком большой или имеет неправильную форму, он может создавать зоны с неравномерным распределением энергии, что влияет на эффективность отражения.
- Точное размещение облучателя относительно рефлектора критично. Если облучатель расположен не в фокусе параболоида, то излучение не будет эффективно сконцентрировано, что приведет к потере части энергии и снижению общего коэффициента эффективности.
- Облучатель должен быть широкополосным, чтобы покрывать все возможные частоты, используемые для коммуникации.

Современные системы беспроводной связи требуют высокой скорости передачи данных, надежности и эффективности использования частотного спектра. Для достижения этих целей используются различные схемы передачи сигналов, включая SISO (Single Input Single Output) и MIMO (Multiple Input Multiple Output). Эти схемы передачи значительно отличаются по архитектуре, производительности и областям применения.

SISO — это традиционная схема передачи сигнала, при которой передача осуществляется через одну антенну на передающей стороне и одну антенну на принимающей стороне. Это наиболее простой и базовый метод передачи данных в беспроводных системах. В SISO-системе передатчик модулирует данные на одну несущую частоту и передает их через одну антенну. Принимающая антенна получает сигнал, который подвергается демодуляции для извлечения переданной информации.

Преимущества SISO:

- SISO-системы требуют меньшего количества аппаратных средств (одна антenna и один радиочастотный тракт), что уменьшает стоимость и сложность системы.
- Одна передающая антenna требует меньше энергии по сравнению с MIMO-системами.
- Обработка сигналов в SISO-системах проще, что уменьшает требования к вычислительным ресурсам.

## Недостатки SISO:

- Так как данные передаются через одну антенну, пропускная способность ограничена скоростью передачи сигнала одной антенны.
- При многолучевом распространении сигнала могут возникать значительные потери из-за интерференции сигналов, что снижает надежность передачи.
- Ограниченнная возможность увеличения скорости передачи данных по сравнению с более сложными схемами, такими как MIMO.

MIMO — это схема передачи, при которой используются несколько антенн как на передающей, так и на принимающей стороне. Эта схема позволяет одновременно передавать несколько потоков данных, увеличивая скорость передачи и улучшая качество сигнала. Основная идея MIMO заключается в использовании пространственного мультиплексирования и/или обработки сигналов для улучшения спектральной эффективности и устойчивости к замираниям. В MIMO-системах можно передавать несколько потоков данных одновременно через различные антенны, что увеличивает пропускную способность без увеличения частотного диапазона.

## Преимущества MIMO:

- Возможность одновременной передачи нескольких потоков данных позволяет значительно увеличить скорость передачи.
- Использование нескольких антенн улучшает устойчивость к замираниям и интерференции, так как каждый поток данных проходит через независимый канал.
- MIMO позволяет более эффективно использовать доступный частотный спектр.

## Недостатки MIMO:

- MIMO-системы требуют дополнительных аппаратных средств (несколько антенн и радиочастотных трактов), что увеличивает стоимость и сложность разработки и развертывания.
- Наличие нескольких передающих и принимающих антенн увеличивает энергозатраты.
- Для обработки сигналов и устранения интерференции требуется более мощное оборудование и более сложные алгоритмы обработки сигналов.

Модемы и системы связи, такие как 4G LTE и 5G, используют адаптивное MIMO, что означает динамическое переключение между режимами приема и передачи сигнала на основе текущих условий связи. Режимы передачи сигнала в MIMO:

- Пространственное мультиплексирование (Spatial Multiplexing): передача нескольких независимых потоков данных через разные антенны, что увеличивает общую скорость передачи данных. Сигнал от каждой антенны не распространяется по прямой линии. В реальной среде сигнал отражается от множества объектов, таких как стены, здания, автомобили и деревья. Это приводит к тому, что один и тот же сигнал проходит по множеству различных траекторий (или путей), прежде чем достигнет принимающей антенны. Эти отраженные сигналы могут иметь разную фазу, амплитуду и задержку из-за разной длины пути. Несмотря на то, что антенны на передающей стороне могут находиться довольно близко друг к другу (например, в пределах нескольких сантиметров в модеме), среда вокруг антенн играет ключевую роль. Каждый сигнал, переданный с отдельной антенны, будет сталкиваться с различными объектами и неоднородностями в окружающей среде, что приведет к возникновению различных путей многолучевого распространения. Таким образом, хотя антенны находятся рядом, пути, по которым распространяются их сигналы, могут значительно различаться. Когда эти разные пути приходят к принимающим антеннам, они имеют разные временные задержки, фазы и уровни сигнала. Принимающая система (например, в смартфоне или базовой станции) использует сложные алгоритмы обработки сигналов, чтобы различить эти потоки данных и восстановить исходные переданные данные. Пространственное мультиплексирование эффективно работает в условиях, когда существует достаточно выраженное многолучевое распространение, и каждый поток данных проходит через различающиеся траектории. В городской среде с множеством отражений от зданий и других объектов это условие обычно выполняется, даже если антенны находятся на расстоянии нескольких сантиметров.
- Разнесенная передача (Diversity Gain): использование нескольких антенн для передачи одного и того же сигнала, но с разными характеристиками, чтобы уменьшить вероятность потери сигнала из-за замираний. В отличие от пространственного мультиплексирования, цель разнесенной передачи — не увеличить скорость передачи данных, а улучшить устойчивость системы к потере сигнала. В MIMO-системах с разнесенной передачей каждая из передающих антенн передает один и тот же поток данных, но с определенными различиями (например, с разными задержками или фазами). Это делается для того, чтобы сигналы прошли через разные траектории в пространстве, и хотя бы один из них достиг приемника в хорошем качестве.
- Формирование луча (Beam forming): направленная передача сигнала с использованием фазовых сдвигов на передающих антенах для концентрации мощности сигнала в определенном направлении.

Выбор между режимами зависит от качества канала связи, и модем принимает решение на основе текущих условий радиоканала с использованием специальных алгоритмов адаптации:

1. Оценка качества канала (Channel State Information, CSI)
  - Мощность сигнала (RSSI — Received Signal Strength Indicator)
  - Коэффициент ошибок (BER — Bit Error Rate)
  - Отношение сигнал/шум (SNR — Signal-to-Noise Ratio)
  - Многолучевые искажения (или глубина замираний)
  - Уровень интерференции (влияние других сигналов)

Эти параметры передаются на базовую станцию или устройство приема через специальные служебные каналы, и на основе этой информации модем принимает решение о том, какой сценарий передачи использовать.

2. Выбор между пространственным мультиплексированием и разнесенной передачей
  - Пространственное мультиплексирование используется, если качество канала хорошее. Это означает, что между передающими и принимающими антеннами существуют независимые пути распространения (разные многолучевые траектории), и каждый поток данных может быть передан без существенных искажений. В этом случае можно одновременно передавать несколько потоков данных, что увеличивает общую пропускную способность.
  - Разнесенная передача используется, когда качество канала среднее или плохое. Если сигнал подвергается сильным замираниям или интерференции, модем переключается на разнесенную передачу, чтобы улучшить надежность связи. В этом случае вместо передачи нескольких потоков данных одновременно, модем будет передавать один и тот же поток через несколько антенн. Это помогает компенсировать возможные потери, так как принимающая сторона сможет выбрать лучший сигнал из нескольких полученных копий.

Миниатюрные и недорогие стик-модемы, не оснащенные внешними коаксиальными разъемами для антенн, почти наверняка относятся к типу SISO. В рамках тестирования нами был приобретен один из таких [модемов](#), см. Рис. 13, на маркетплейсе Wildberries. USB-разъем используется исключительно для питания модема. Его можно подключить и к зарядному устройству, где ток должен быть не менее 1.5 А, иначе модем не будет функционировать. После запуска модема его настройка осуществляется через веб-интерфейс в браузере, доступный по Wi-Fi соединению. Адрес для подключения, который необходимо ввести в адресную строку браузера, указывается производителем в инструкции. В нашем случае это был 192.168.100.1. На открывшейся веб-странице необходимо ввести логин и пароль (который можно изменить), после чего можно выполнить настройки.

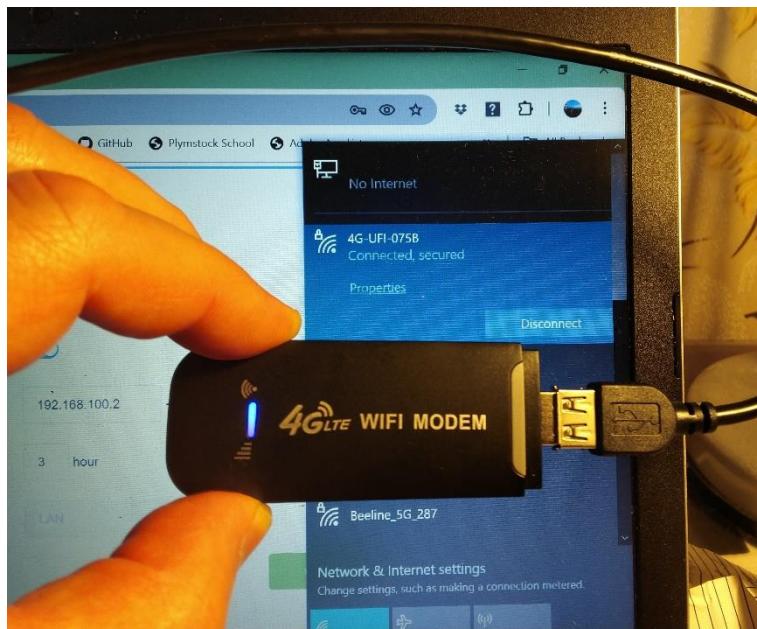


Рис. 13. 4G стик-модем без внешних разъемов для антенны.

От использования такого модема следует сразу отказаться по нескольким причинам: нестабильность работы, низкое качество исполнения, и невозможность подключения внешней антенны (понадобится на следующих этапах проекта). Во-первых, было установлено, что модем обладает крайне нестабильным Wi-Fi, который отключается с четкой периодичностью. Затем, после нескольких подсоединений, USB-разъем стал шататься. Чтобы выяснить причину, мы вскрыли крышку модема и обнаружили на печатной плате так называемую холодную пайку — ситуацию, когда при пайке не произошла адгезия между контактом и припоеем, как показано на Рис. 14. Нам пришлось перепаять весь USB-разъем.

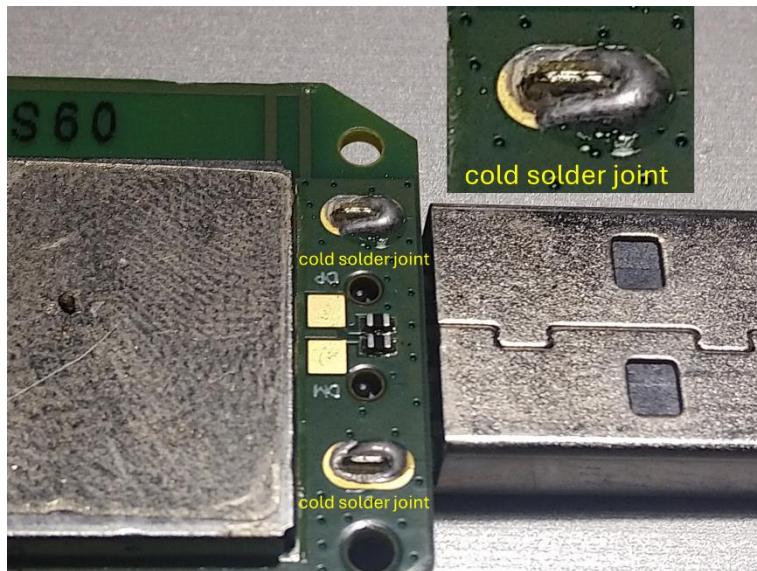


Рис. 14. Пример «холодной пайки» на печатной плате стик-модема. Видно, что пайка не обволакивает контакты.

При вскрытии корпуса модема были обнаружены лепестковые контакты, предназначенные для единственной антенны (см. Рис. 15), расположенной на пластмассовом колпачке. Это подтверждает наше предположение о том, что модем относится к типу SISO. Принцип работы антенны не удалось определить. При измерениях с помощью мультиметра оказалось, что контакты «GND» и «ANT» находятся в коротком замыкании, что вызвало удивление. На внутренней стороне колпачка видна квадратная пластина (см. Рис. 15), которая, возможно, выполняет функцию патч-антенны, однако оба лепестка касаются ее поверхности. Назначение черной накладки на колпачке в виде липкой ленты также осталось неясным. Таким образом, вопрос о конструкции антенного входа остается открытым.

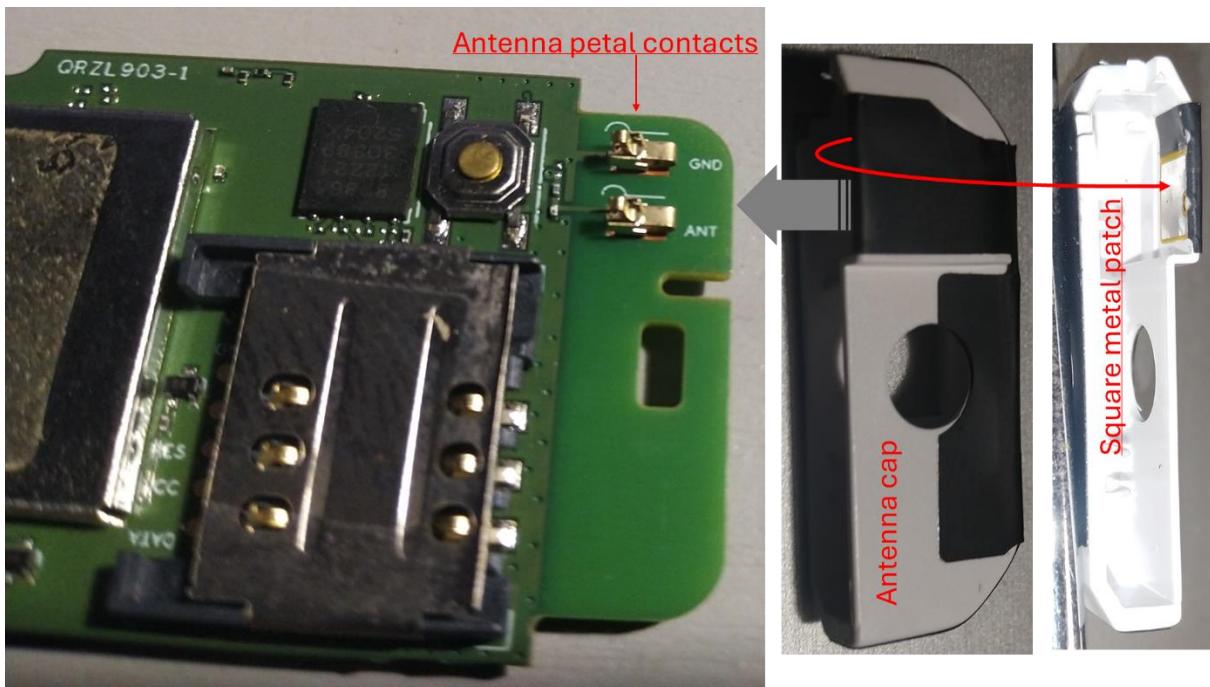


Рис. 15. Внутренняя SISO антenna модема.

Нашей второй попыткой стал 4G стик-модем SISO-типа от оператора связи [МегаФон](#) (см. Рис. 16), с использованием его SIM-карты. Мы предварительно убедились, что оператор МегаФон предоставляет 4G покрытие на территории нашего СНТ. Кроме того, на сайте оператора указано наличие базовой станции в селе Александрово, на которую мы наводили рефлектор. Этот модем представляет собой совершенно иной уровень исполнения, включая удобный интерфейс в браузере, однако его стоимость в четыре раза выше по сравнению с предыдущим устройством. Хотя вся эта электроника производится в Шэньчжэне (Китай), ключевую роль играет бренд, который заказывает продукцию, а значит, и контроль качества. Первый модем был безбрендовым, и в нем экономили на всем, включая компоненты, сборку, и программное обеспечение. МегаФон модем может работать автономно, но в нем также предусмотрен коаксиальный вход для внешней SISO антенны.



**Рис. 16.** 4G стик-модем SISO-типа от оператора связи МегаФон.

На начальном этапе проекта мы предприняли весьма смелую, хотя и технически необоснованную попытку — установили МегаФон стик-модем в фокусе рефлектора, как показано на Рис. 17. Питание модема осуществлялось от зарядного устройства через USB-удлинитель. Понимая принципы работы облучателя (направленной антенны) и рефлектора, данную схему можно подвергнуть серьезной критике. Во-первых, нам совершенно неизвестна диаграмма излучения модема, что затрудняет выбор его оптимальной ориентации в фокусе рефлектора. Во-вторых, эта диаграмма явно не предназначена для работы с рефлектором. Кроме того, поляризация приходящей волны, сфокусированная на модем, может не совпадать с ориентацией внутренней антенны модема, что приведет к дополнительным потерям сигнала и снижению эффективности приема. Из-за этих факторов эффективность системы в целом не может быть высокой (малое значение коэффициента  $k$  в (14)). Тем не менее, нам удалось получить усиление сигнала на уровне 10 dB, хотя при использовании надлежащего облучателя даже с посредственной эффективностью рефлектора ( $k = 0.6$ ) можно было бы ожидать усиление от 17 до 26 dB. На результат, безусловно, также повлияла недостаточная точность воспроизведения поверхности параболоида. Все это будет учтено на следующих этапах проекта.

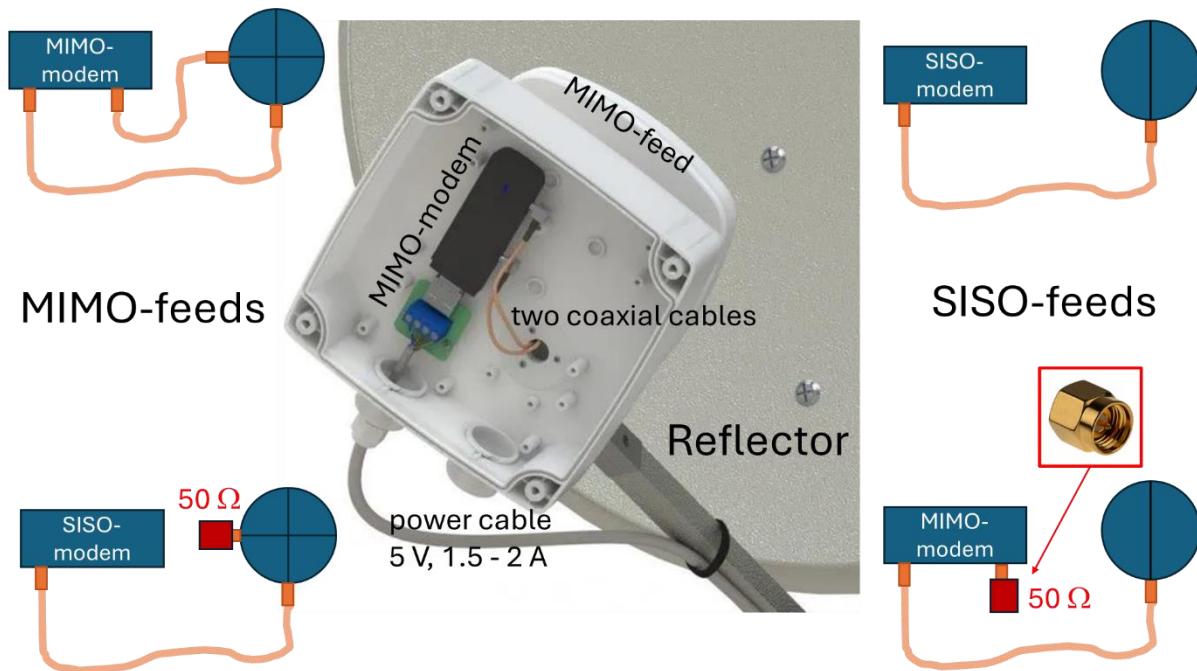
Еще раз отметим, что решение по установке модема в фокусе рефлектора без облучателя было сделано не из-за неверных представлений, а с целью исследования и обоснования комплектации будущей системы. Это включало разработку технологии изготовления рефлектора, выбор модема и облучателя. Читателям нашего отчета, желающим реализовать предложенные решения, уже не придется повторять этот начальный этап.



**Рис. 17.** МегаФон 4G стик-модем, установленный в фокусе рефлектора.

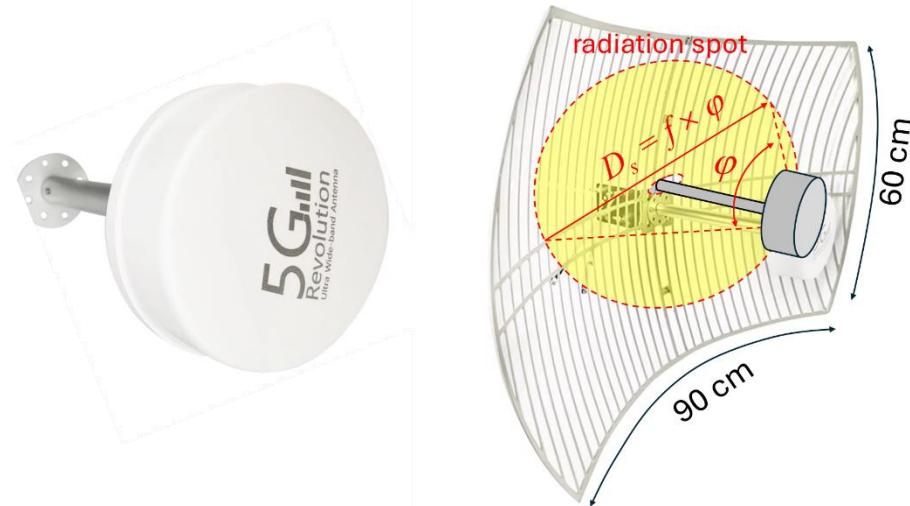
Следующим шагом в улучшении системы с рефлектором должно стать подключение внешнего облучателя типа SISO или MIMO, в зависимости от используемого модема. Оба облучателя представляют собой широкополосные направленные антенны. В SISO-облучателе используется одна антenna, тогда как в MIMO-облучателе предусмотрены две антенны (и, соответственно, два выходных кабеля), с пересекающимися линейными поляризациями для минимизации кросс-толка (взаимного влияния) между ними. Если речь идет о компактном стик-модеме, его можно было бы разместить рядом с облучателем и подключить короткими кабелями, чтобы минимизировать потери при передаче сигналов. Эта схема показана на Рис. 18 для MIMO-модема (уже коммерчески доступное решение). В случае использования модема типа SISO (с одним кабелем) можно применять MIMO-облучатель, однако неиспользованный выход на облучателе (с любой поляризацией) рекомендуется заглушить 50  $\Omega$  коаксиальной нагрузкой (разъем SMA или N-type) с рабочим диапазоном до 6 ГГц, которую легко найти и приобрести.

Тем не менее, схема, представленная на Рис. 18, не кажется нам полностью оптимальной. Дело в том, что в фокусе рефлектора, усиливающего сигналы, находятся разнородные источники излучения, такие как облучатель и сам модем, что может негативно отразиться на работе MIMO-каналов связи при передаче данных из-за интерференции, однако точное воздействие нам неизвестно. Возможно, что схема с SISO-облучателем окажется менее чувствительной к особенностям конфигурации системы. В связи с указанными сомнениями представляется целесообразным разместить модем вне зоны облучения рефлектора, например, позади него. При этом коаксиальные кабели останутся достаточно короткими, не более метра. Примерами 4G модемов MIMO-типа, которые можно использовать для такой конфигурации, являются [ZTE MF283](#) (поддерживает до 32 абонентов) или [Huawei B818-263](#) (поддерживает до 64 абонентов). Оба модема оснащены двумя коаксиальными разъемами для подключения внешних антенн. Модем Huawei обладает большей [функциональностью](#), предоставляя возможность использования [VPN-соединения](#), LAN и телефонный разъемы, а также сохраняет PIN-код SIM-карты после отключения питания. Эта функция особенно важна для использования на даче, где возможны перебои с электричеством. В противном случае, как в случае с модемом МегаФон, пришлось бы каждый раз заново заходить в настройки устройства в браузере и вводить PIN-код (0000 или 1111, если он не был изменен на уникальный).



**Рис. 18.** Схема инсталляции стик-модема совместно с облучателем, а также различные схемы соединения модема и облучателя в зависимости от их типов.

При выборе облучателя, представляющего собой антенну направленного действия, важно учитывать рабочий диапазон частот, чтобы он полностью охватывал частоты, используемые вашим оператором для 4G связи. Широкополосный облучатель ММО-типа, найденный нами на [AliExpress](#), имеет диапазон от 698 до 6000 МГц, что позволяет использовать его с любым модемом. Для подключения кабелей к модему могут понадобиться коаксиальные адаптеры, которые также можно легко приобрести на AliExpress. Этот облучатель поставляется с металлической крепежной штангой, которая задает рекомендованное фокусное расстояние для усеченного сетчатого рефлектора размером 90 × 60 см<sup>2</sup>, с которым он обычно используется, как показано на Рис. 19.

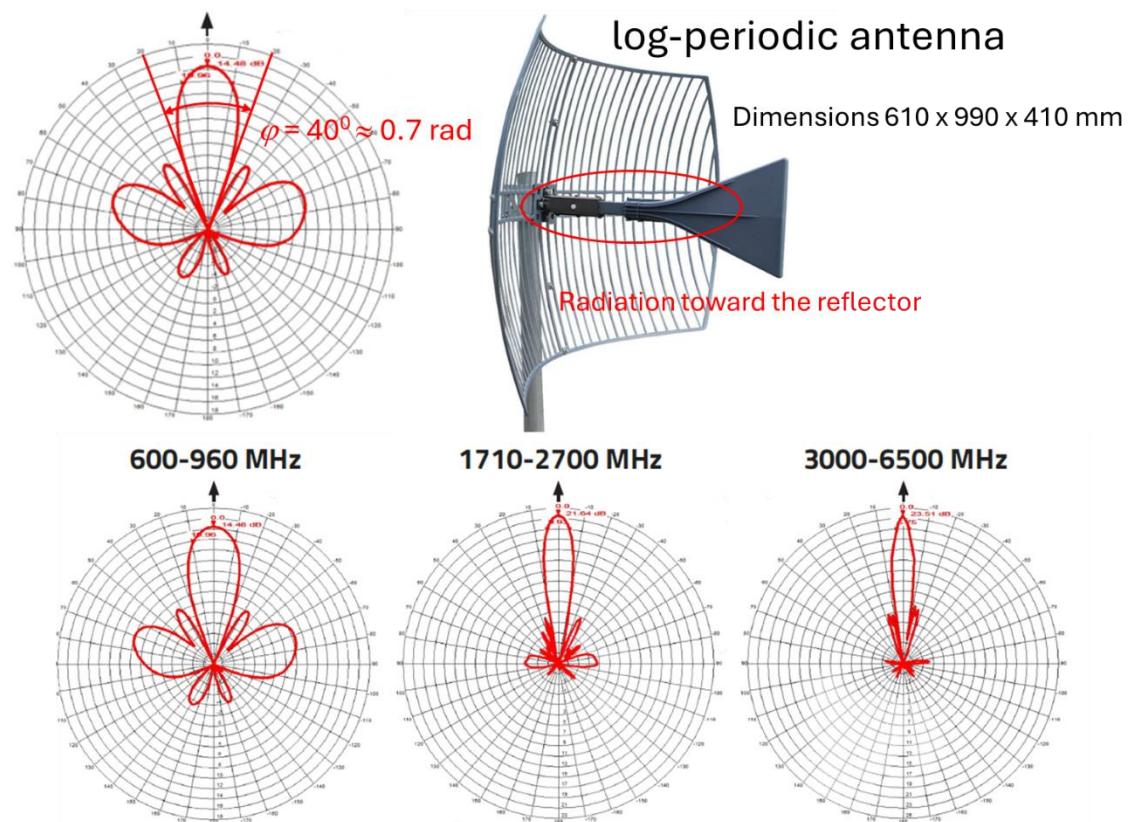


**Рис. 19.** Пример широкополосного ММО-облучателя, установленного на сетчатый рефлектор.

Первый вопрос, который возникает в связи с этим облучателем, — в каком направлении он излучает? Производитель не предоставил никакой информации (Китай, бренд отсутствует), и мы также не знаем деталей его конструкции. Обычно облучатели направляют сигнал в сторону рефлектора для передачи и принимают отраженный и сфокусированный сигнал обратно. Однако вызывает сомнение металлическая крепежная штанга, исходящая из центра корпуса облучателя, которая может создавать помехи для излучения. Тем не менее, будем исходить из того, что используется классическая схема, при которой облучатель излучает в сторону рефлектора. Облучатель формирует диаграмму направленности, включающую основной луч вдоль оси, а также боковые и обратные лучи, которые не участвуют в передаче сигнала. Коэффициент усиления облучателя определяется отношением энергии, излучаемой в основном направлении, к энергии, излучаемой в другие стороны. На поверхности рефлектора основной луч с углом раскрыва  $φ$  (в радианах) формирует « пятно » диаметром  $D_s = f \times φ$ , где  $f$  — фокусное расстояние. Размер пятна зависит от частоты: чем ниже частота, тем больше угол раскрыва. Для эффективной работы рефлектора « пятно » должно находиться в его пределах при любой частоте, чтобы избежать ситуации, когда облучатель излучает за пределы рефлектора. Это требование и определяет минимальный размер рефлектора.

Поэтому при изготовлении рефлектора очень важно знать максимальный угол раскрыва основного луча на минимальной частоте рабочего диапазона облучателя. В описании облучателя производитель указал горизонтальные и вертикальные углы главного луча —  $9^0$  и  $8^0$  соответственно. Это очень узкий луч (слегка несимметричный). Поскольку, как мы отметили выше, эти углы зависят от частоты, скорее всего, они относятся к максимальной частоте 6 ГГц. Для нижней частоты 698 МГц они будут значительно шире. Не обладая достоверными данными, можно, тем менее, оценить максимально возможный угол раскрыва основного луча (в радианах), поделив минимальный размер рефлектора 60 см на длину штанги (фокусное расстояние).

На Рис. 20 мы показываем еще одно интересное решение для облучателя от компании [Waveform](#). На AliExpress можно найти [китайские безбрендовые аналоги](#), которые гораздо дешевле. Насчет их качества и соответствия заявленным характеристикам мы ничего не можем гарантировать. [Технические характеристики](#) антенны и [руководство по его установке](#) доступны на веб-сайте компании (на английском). Облучатель представляет собой широкополосную (600 – 6500 MHz) SISO-антенну (один кабель) с узкой диаграммой направленности в сторону рефлектора.

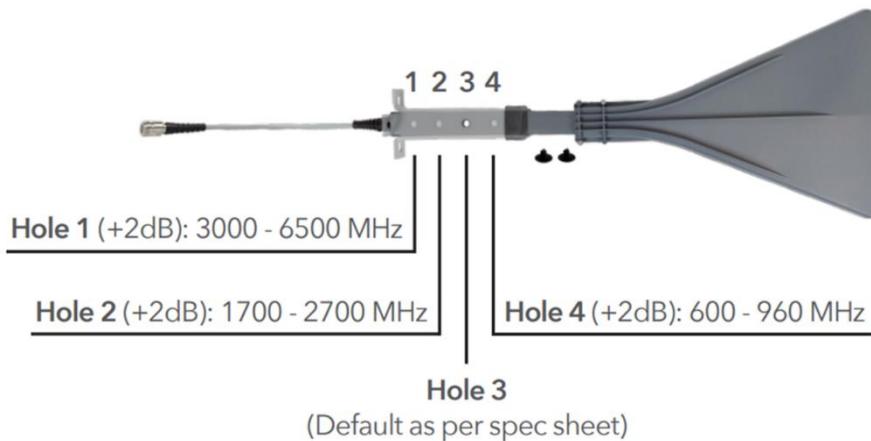


**Рис. 20.** Широкополосный SISO-облучатель, установленный на сетчатый рефлектор. Диаграммы направленности (красные) относятся только к облучателю (логарифмически-периодической антенне), а не всей системе в целом.

Тип антенны (скрытой в защитном кожухе) нам удалось узнать непосредственно от инженеров компании – это так называемая логарифмически-периодическая антенна ([log-periodic antenna](#)). Возможно, она исполнена на печатной плате – этого мы точно сказать не можем. Но вообще этот тип антенн хорошо описан в литературе, поэтому ее в принципе можно изготовить самому. Заметьте, что эта антenna также крепится на центральной металлической штанге, как и предыдущий облучатель. Однако это не вызвало возражений у дизайнеров, которым мы склонны доверять.

Эта антenna светит в сторону сужающегося конца, т.е. в сторону рефлектора. Из диаграмм направленности, которые были [предоставлены](#) компанией (см. Рис. 20), можно оценить максимальный размер « пятна » облучения на рефлекторе, взяв максимальный угол раскрыва основного луча  $\varphi \approx 0.7$  rad и длину антены 410 мм:  $D_s = 0.7 \times 410 \approx 290$  мм, что в два раза меньше вертикального размера рефлектора 610 мм. Так и должно быть – размер рефлектора должен превышать размер максимального « пятна ».

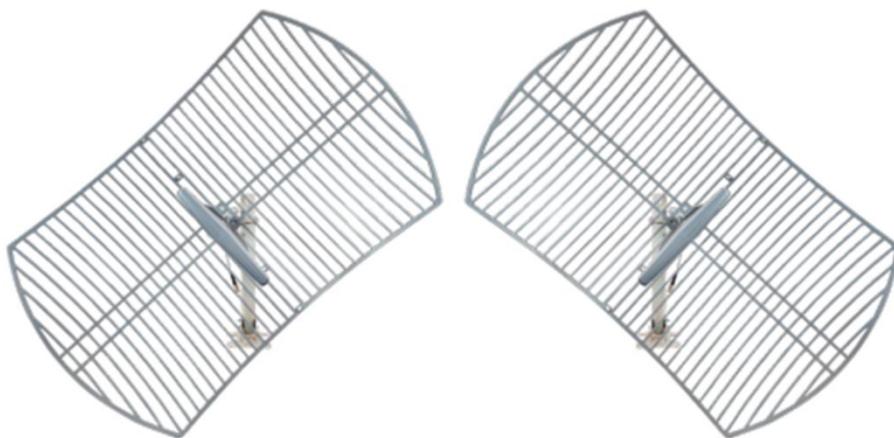
Штанга, на которой крепится антена, имеет четыре фиксированных позиции вдоль оси, см. Рис. 21. Как объясено в [руководстве по установке](#), меняя позицию, можно увеличить усиление на 2 dB для выбранного диапазона. Если это незначительное улучшение не принципиально для вас или вы не знаете рабочий диапазон, то рекомендуется установить штангу в позицию «3». Само наличие такой опции говорит о том, что антена была тщательно продумана и протестирована. Это внушает доверие к ее заявленным характеристикам.



**Рис. 21.** Регулировка усиления антенны для выбранного диапазона путем перемещения ее на фиксированные позиции на штанге.

В MIMO-облучателе, где имеются две антенны со скрещенными поляризациями, поляризация приходящих и сфокусированных электромагнитных волн будет в любом случае совпадать с одной из антенн или частично с обеими. Это означает, что независимо от поляризации пришедшей волны, она будет эффективно приниматься по крайней мере одной из антенн. Такая конфигурация снижает чувствительность к ориентации антенны вокруг ее оси и улучшает общую эффективность приема сигнала. Тогда SISO-облучатель с одной антенной может оказаться более чувствительным к ориентации антенны вокруг ее оси. С другой стороны, SISO-облучатель с одной антенной фиксированной поляризации более чувствителен к ориентации антенны вокруг ее оси. Если поляризация приходящей волны не совпадает с поляризацией антенны, эффективность приема сигнала может снизиться. Поэтому для SISO-облучателей правильная ориентация важна для максимизации качества приема. Для рассматриваемого нами облучателя его независимое вращение вокруг оси не предусмотрено, но его можно поворачивать вместе с рефлектором во время настройки на вышку связи.

В [руководстве по установке](#) также обсуждается использование двух скрещенных рефлекторов, см. Рис. 22, для подключения к MIMO-модему. Такая конфигурация воспроизводит скрещенные поляризации в MIMO-облучателях. Схема с двумя рефлекторами и SISO-облучателями заслуживает внимательного изучения, и мы рассматриваем ее как весьма перспективную для нашего проекта.



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

Из нашего краткого обзора по MIMO-модемам ([см. стр. 24](#)) мы знаем, что существует три режима передачи данных, из которых только один — пространственное мультиплексирование, обеспечивающее повышенную скорость передачи. В остальных режимах модем «борется» за стабильность канала, а не скорость передачи. Чтобы перевести модем в этот режим, необходимо (1) обеспечить высокий уровень сигнала и (2) предоставить два некоррелированных канала передачи данных. При использовании одного MIMO-облучателя с близко расположеннымами внутренними антеннами установление двух различных каналов может не произойти, тогда как в схеме с двумя рефлекторами это более вероятно. Таким образом, схема с двумя рефлекторами и SISO-облучателями

действительно может быть более эффективной для реализации режима пространственного мультиплексирования в ММО-системах. Это связано с уменьшением корреляции между каналами и улучшением условий для передачи независимых потоков данных.

Вне зависимости от выбранного облучателя мы будем изготавливать рефлектор собственной конструкции и гораздо большего диаметра (до 1.5 м). При увеличении диаметра рефлектора  $D_r = 2R$  (см. (2)) необходимо сохранить отношение  $f/D_r$  рекомендованное для данного облучателя, где  $f$  – фокусное расстояние. Обычно оно лежит в пределах 0.5 – 0.9 (в нашем картонном рефлекторе 0.5). Добросовестные производители облучателей будут указывать рекомендуемое отношение  $f/D_r$ . Обращайте внимание на рабочий диапазон, который должен полностью охватывать весь спектр 4G, и убедитесь, что облучатель имеет импеданс 50 Ом для согласования с модемом (вся микроволновая техника обычно рассчитана на 50 Ом).

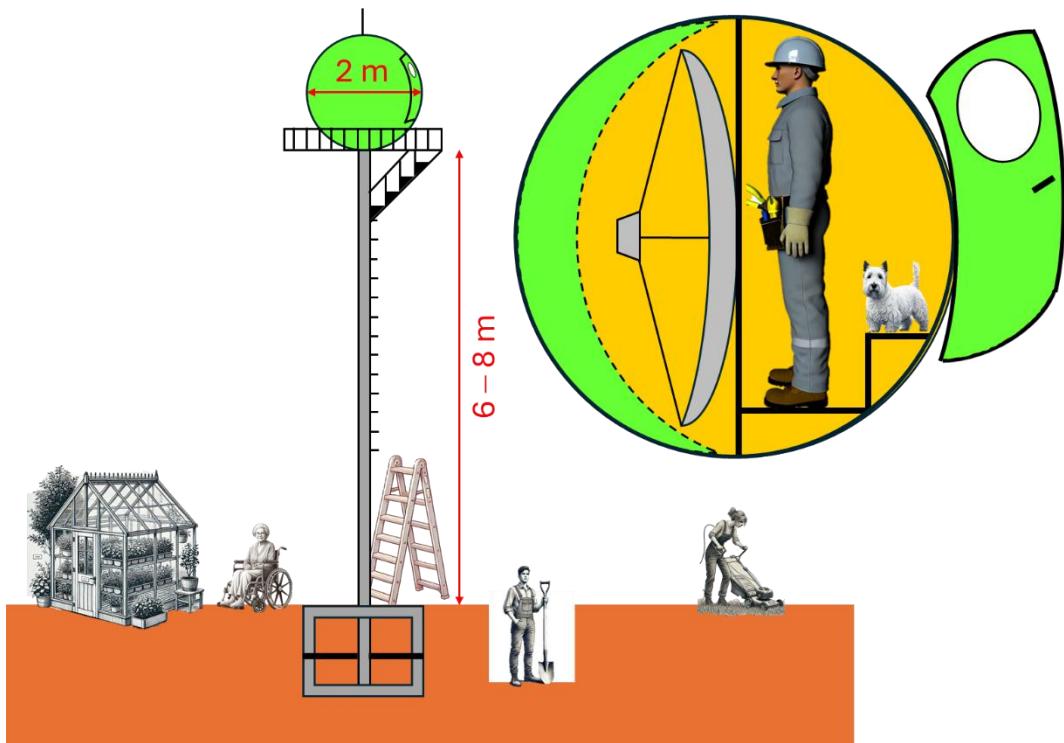
Мы также заинтересованы в разработке собственных облучателей на основе хорошо зарекомендовавших себя схем, опубликованных в книгах и научных журналах. Доступ к такой литературе у нас имеется. В данном отчете мы не касаемся теории антенн и электроники, которые потребуются для создания продвинутых рефлекторов или целой приемной станции. Все это — темы для следующего отчета. Однако уже на основе настоящего отчета можно начать изготовление собственного рефлектора и выбрать подходящие модем и облучатель, доступные в продаже.

## Общие принципы построения мачтовой станции приема

Несколько факторов заставляют задуматься о необходимости установки мачтовой станции для приема сигнала. Все уже усвоили, что антенну нужно поднимать как можно выше, что особенно актуально в условиях лесного массива. В принципе, рефлектор, даже большого диаметра, можно разместить на чердаке, как в нашем случае, если крыша неметаллическая (шифер, ондулин). Однако в ряде случаев такое размещение неприемлемо — например, из-за жилых помещений на чердаке или нехватки места. Тогда рефлектор придется выносить наружу. Поскольку это не спутниковый рефлектор, направленный в небо, потребуется высокое расположение на фронтоне дома, ориентированное в сторону вышки связи. Но если ориентация дома не позволяет такое размещение, возникают дополнительные трудности. Установка рефлектора на обычную трубную мачту также не подходит из-за его большой парусности и сложных погодных условий (вспомним ледяные дожди зимой). Ржавые рефлекторы на фасадах домов уже стали привычным явлением, но это, как правило, небольшие спутниковые тарелки, более устойчивые к воздействию окружающей среды. Какое решение можно предложить в таком случае?

Если установка рефлектора на чердаке или фронтоне невозможна, либо вызывают опасения погодные условия при фронтонной установке, то остается один вариант — возведение отдельно стоящей мачты. На ее вершине следует разместить защитную сферу, которая вместит рефлектор,

модем и другое необходимое оборудование. Эскиз мачты представлен на Рис. 23. Ее ствол, высотой 6 – 8 м, может состоять из секций стальных труб диаметром 200 мм. Вариант деревянной мачты из цельного толстого ствола тоже подходит. Насчет фундамента определитесь сами, исходя из вашего опыта и бюджета. Это может быть, например, колодезное кольцо (см. Рис. 23). На вершине сферы необходимо установить громоотвод.



**Рис. 23.** Мачта с рефлектором внутри защитной сферы.

Диаметр сферы должен слегка превышать диаметр рефлектора. Она может быть изготовлена из тонких фанерных лепестков по аналогии с технологией создания рефлекторов, с последующим покрытием водоотталкивающим лаком для защиты от атмосферных воздействий и влаги. Для дополнительной защиты поверхность сферы может быть покрыта стеклотканью с эпоксидной смолой, которая после полимеризации образует прочный защитный слой. Альтернативно, фанерная сфера может быть использована в качестве формы для изготовления полностью композитной сферы методом оттиска, что позволит получить конструкцию с высокой прочностью при малом весе. Алгоритм раскроя лепестков приведен в следующем разделе отчета.

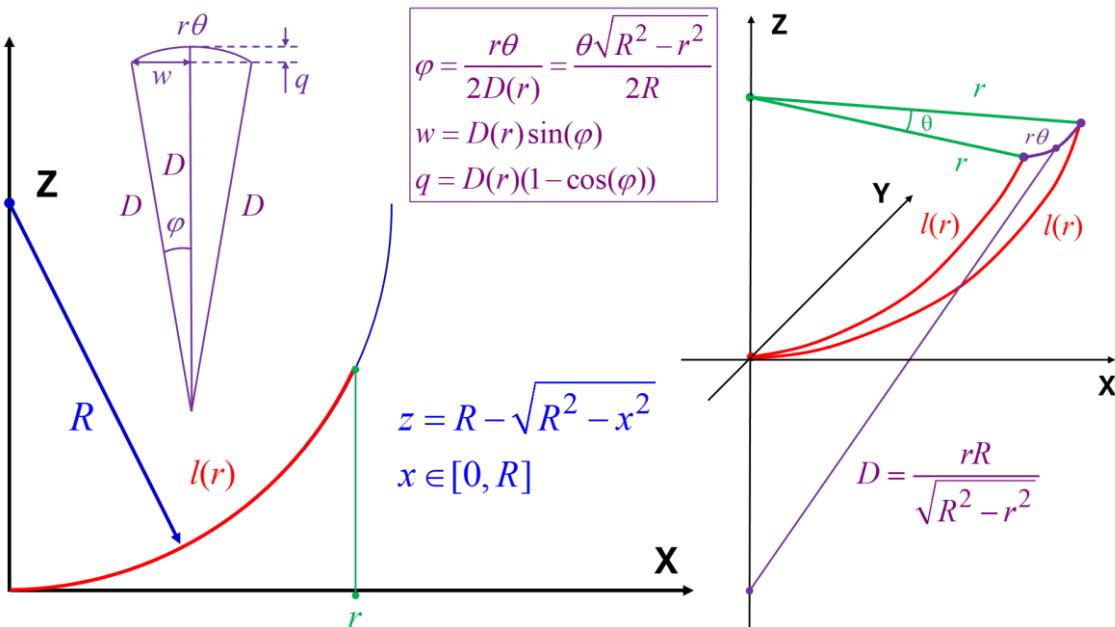
В настоящем отчете мы ограничимся лишь кратким обсуждением вопросов, связанных с комплектованием станции. Ее оснащение будет зависеть от ваших задач и квалификации. Поскольку рефлектор займет только половину внутреннего пространства сферы, остается достаточно места для размещения дополнительного электронного оборудования, помимо модема. Мы планируем оснастить рефлектор электромеханической системой для точного наведения на вышку связи, а также актуаторами, которые позволят вращать облучатель вокруг его оси и изменять фокусное расстояние

для оптимизации поляризации и усиления сигнала. Также было бы полезно установить бесперебойный источник питания. В сфере будет находиться мини-компьютер [Raspberry Pi](#), который будет подключен к модему через LAN-кабель. С его помощью можно управлять через Wi-Fi всеми периферийными устройствами, используемыми для автоматизации на участке.

Необходимо также продумать систему вентиляции внутри сферы, так как летом там, очевидно, будет очень жарко. На чердаке дома тоже жарко, но его легче проветривать. Что касается работы в зимний период, то здесь требуется дальнейшая проработка. Для круглогодичного функционирования электронных приборов понадобится электрическое отопление внутри сферы, хотя температурные требования будут значительно ниже, чем в жилом помещении. Однако, куда более значительной проблемой может стать влажность.

### Раскройка лепестков для изготовления сферы

Рассмотрим четверть окружности  $z(x) = R - \sqrt{R^2 - x^2}$ , с радиусом  $R$  и  $x \in [0, R]$ , в системе координат на Рис. 24. Вращая эту кривую вокруг оси Z, мы получим полусферу. Принцип раскройки полусферы останется прежними, как и для параболоида, но изменятся формулы, а также при численных расчетах необходимо устраниить неопределенности типа  $0 \times \infty$ , как объяснено ниже.



**Рис. 24.** Геометрические параметры для расчета лепестков, из которых будет скроена сфера.

Каждому радиусу  $r \in [0, R]$ , откладываемому от оси вращения Z, будет соответствовать определенная длина окружности  $l(r)$ :

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

При общем диаметре сферы  $D_s = 2R$ , мы получаем для длины  $L$  каждого лепестка:

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

Для определения профиля лепестка нам понадобится определять значение  $r$  по заданному значению  $l$  вдоль окружности:

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

По сравнению с параболой, здесь нам не потребуется решать нелинейное уравнение. Для величины касательного отрезка  $D$ , см. Рис. 24, получаем:

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

Таким образом,  $D$  становится бесконечным на экваторе сферы, когда  $r = R$ . В случае параболы такого не происходило ни при каком конечном  $r$ . В силу этого необходимо слегка модифицировать алгоритм, чтобы исключить отдельное вычисление  $D$ , которое может принимать сколько угодно большие значения и привести к переполнению арифметики при численных расчетах. Также, необходимо устранить неопределенности типа  $0 \times \infty$ , которые возникают в формулах для  $w$  и  $q$ . Для этого мы воспользуемся следующими разложениями в ряд Тейлора при  $\varphi \rightarrow 0$ :

$$\sin(\varphi) = \varphi + O(\varphi^2) \quad (19)$$

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

Принимая во внимание (19) и (20), запишем уравнения для  $w$  и  $q$  в следующей эквивалентной форме:

$$w = D(r) \sin(\varphi) = D(r)(\sin(\varphi) - \varphi) + D(r)\varphi \quad (21)$$

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

Используя (18) и  $\varphi = \frac{\theta \sqrt{R^2 - r^2}}{2R}$  (см. Рис. 24), получаем:

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

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

В (23) и (24) уже не возникает неопределенностей, поскольку  $(\sin(\varphi) - \varphi)$  и  $\left( 1 - \cos(\varphi) - \frac{\varphi^2}{2} \right)$  в чисителях убывают гораздо быстрее при  $r \rightarrow R$ , чем знаменатель  $\sqrt{R^2 - r^2}$ . Поэтому не возникнет переполнения арифметики при вычислении этих отношений.

Теперь мы готовы изложить алгоритм раскройки сферы:

1. Вычислить угловую ширину лепестка  $\theta = 2\pi/N$ , где  $N$  число лепестков.
2. Для заданного  $R$  вычислить общую длину лепестка  $L$  по формуле (16).
3. Вычислить ряд длин лепестков  $l_i = iL/(M - 1)$ ,  $i = \overline{0, M - 1}$ , где  $M$  достаточное число разбиений, определяющих точность построения профиля. Нумерация ведется от  $i = 0$ , как это принято в программировании.
4. Вычислить ряд радиусов  $r_i$ , соответствующих  $l_i$  по формуле (17),  $i = \overline{0, M - 1}$ .
5. Вычислить ряд углов раскрыва лепестков  $\varphi_i = \frac{\theta \sqrt{R^2 - r_i^2}}{2R}$ ,  $i = \overline{0, M - 1}$ .
6. Вычислить ряд смещений  $q_i$  по формуле (24),  $i = \overline{0, M - 1}$ .
7. Вычислить ряд меридианных координат  $s_i = l_i - q_i$ ,  $i = \overline{0, M - 1}$ .
8. Вычислить ряд ширин  $w_i$  по формуле (23),  $i = \overline{0, M - 1}$ , откладываемых от соответствующих меридианных координат  $s_i$  в обе стороны.
9. Для замыкания лепестка соединить концы конечных ширин  $w_{M-1}$  прямой линией (бесконечный радиус кривизны замыкающей дуги).

Программный код на Python для расчета профиля лепестка сферы приведен ниже. Он может быть скачан из [репозитария проекта](#) (файл Sphere.py). Программа представляет собой консольное приложение, поэтому не имеет пользовательского интерфейса. Все параметры (Design parameters) необходимо ввести непосредственно в текст программы. В результате исполнения программы создается табличный файл Spherical\_profile\_data.csv, столбцы которого содержат все дискретные параметры, вычисляемые в алгоритме. Для построения профиля лепестка ( $s_i, w_i$ ) понадобятся только два последних столбца  $s$  и  $w$ .

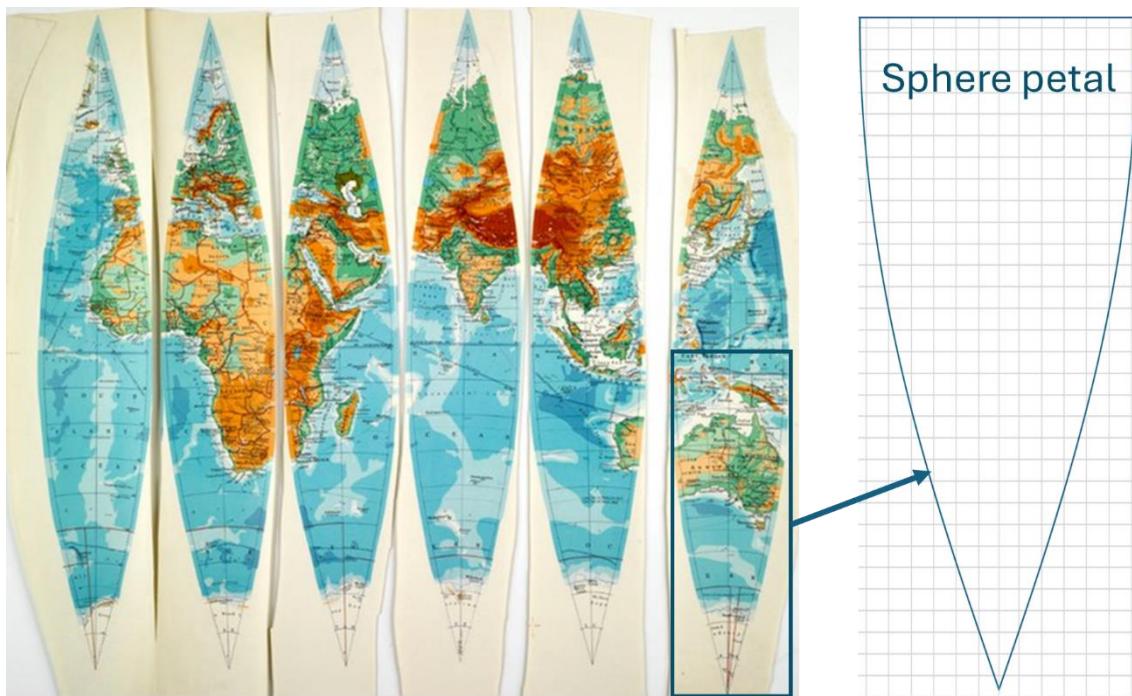
```

1. #
2. # Algorithm for cutting the petals for a sphere
3. #
4. # Dmitriy Makhnovskiy, September 2024
5. #
6.
7. import csv
8. import numpy as np
9.
10. pi = np.pi # pi-constant 3.1415....
11.
12. # Design parameters
13. R = 1000.0 # dish radius; your units (mm, cm, or m)
14. N = 10 # number of petals used for the sphere
15. M = 50 # number of points on the petal template for drawing its profile
16.
17. # Calculated parameters
18. theta = 2.0 * pi / N # angular width of the petal
19. L = pi * R / 2.0 # petal length
20.
21. l = [0.0] * M # points along the petal
22. l[M - 1] = L
23. r = [0.0] * M # array of r corresponding to the array of l

```

```
24. r[M - 1] = R
25. eff = [0.0] * M # array of the opening angles
26. eff[M - 1] = 0.0
27. q = [0.0] * M # meridian displacements with respect to the l-points
28. q[M - 1] = 0.0
29. s = [0.0] * M # array of the meridian coordinates
30. s[M - 1] = L
31. w = [0.0] * M # array of the widths corresponding to the array of s
32. w[M - 1] = theta * R / 2.0
33.
34. for i in range(0, M-1):
35.     length = i * L / (M - 1)
36.     l[i] = length
37.     r[i] = R * np.sin(length / R)
38.     sqroot = np.sqrt(R ** 2 - r[i] ** 2)
39.     eff[i] = theta * sqroot / (2.0 * R)
40.     q[i] = r[i] * (1.0 - np.cos(eff[i])) - eff[i] ** 2 / 2.0 / sqroot + r[i] * theta ** 2 * sqroot / (8.0 *
R)
41.     s[i] = l[i] - q[i]
42.     w[i] = r[i] * (np.sin(eff[i]) - eff[i]) / sqroot + r[i] * theta / 2.0
43.
44. # Saving the profile data to a CSV file
45. data = np.column_stack((l, r, eff, q, s, w))
46. header = ['l', 'r', 'eff', 'q', 's', 'w']
47.
48. with open('Spherical_profile_data.csv', 'w', newline='') as csv_file:
49.     writer = csv.writer(csv_file)
50.     writer.writerow(header)
51.     writer.writerows(data)
```

Пример рассчитанного лепестка для сферы показан на Рис. 25. Как нам хорошо известно из школьной программы, этот же принцип используется при создании географических атласов. Глобус разделяется на вертикальные сегменты, называемые лепестками. Эти сегменты имеют форму вытянутых долек (см. Рис. 25), которые охватывают поверхность от одного полюса к другому. Каждая долька представляет собой часть поверхности глобуса, которая затем будет отображаться на плоском листе. Каждый лепесток глобуса проецируется на плоский лист бумаги или карты. Поскольку сферическая поверхность невозможно идеально развернуть на плоскость без искажений, используются специальные проекции (например, гномоническая, синусоидальная или проекция Меркатора), чтобы минимизировать искажения формы, площади и расстояний. После того как все лепестки глобуса развернуты и проецированы, они объединяются на страницах атласа.



**Рис. 25.** Рассчитанный лепесток сферы и применение сферической раскройки в картографии.

Монтаж лепестков полусфера должна выполняться с прилеганием их швов к жестким ребрам рамы, воспроизводящим четверти окружности. Только таким образом можно обеспечить касательное соединение лепестков, как это было описано для параболоида. Рама может быть расположена как снаружи поверхности, так и внутри нее. Лепестки будет проще притягивать к раме, если она находится снаружи. Готовую поверхность можно многократно использовать для изготовления композитных форм методом оттиска.

### Алгоритмы раскройки с графическим интерфейсом в браузере

На основе наших программных кодов на Python (`Sphere.py` и `Parabolic_reflector.py`), мы сгенерили (используя [ChatGPT-4](#)) коды на [JavaScript](#) с графическим интерфейсом, вызываемым в браузере. Они сохранены в файлах `Sphere.html` и `Parabolic_reflector.html` и доступны для скачивания в [репозитории проекта](#). Для запуска программ, достаточно кликнуть на файл мышкой и позволить вашему браузеру (Google, Microsoft Edge) открыть вызываемую веб-страницу, в окне которой необходимо ввести параметры параболоида или сферы. Во избежание ошибок, при введении параметров перейдите на английскую клавиатуру. Поскольку частоты могут быть десятичными дробями (0.9 ГГц), и используйте точку «.» для введения нецелых значений. Обращаем ваше внимание, что программы будут работать только на компьютере, но не на смартфоне. Использование JavaScript стало возможным из-за отсутствия специфических численных библиотек в наших программах на Python в силу простоты алгоритмов. Таким образом, вам не придется устанавливать громоздкие Python и IDE, чтобы осуществлять расчеты. Интерфейсы в браузере и программные коды показаны ниже.

## Parabolic Reflector Profile with CSV and Text File Generation

Dish Radius (R):

Focus Length (f):

Number of Petals (N):

Number of Points on Petal (M):

Minimum Frequency (f\_min in GHz):

Maximum Frequency (f\_max in GHz):

Reflector Efficiency (k):

Select Units:

Calculation Precision (eps):

### Parabolic\_reflector.html

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.     <meta charset="UTF-8">
5.     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.     <title>Parabolic Reflector Profile</title>
7.     <style>
8.         body {
9.             font-family: Arial, sans-serif;
10.            margin: 20px;
11.        }
12.        label, input, select {
13.            margin: 5px 0;
14.            display: block;
15.        }
16.        button {
17.            margin: 10px 0;
18.        }
19.        #output {
20.            margin-top: 20px;
21.            background-color: #f0f0f0;
22.            padding: 10px;
23.            border: 1px solid #ccc;
24.        }
25.    </style>
26. </head>
27. <body>
28.     <h1>Parabolic Reflector Profile with CSV and Text File Generation</h1>
29.
30.     <label for="R">Dish Radius (R):</label>
31.     <input type="number" id="R" placeholder="Enter R">
32.
33.     <label for="f">Focus Length (f):</label>
34.     <input type="number" id="f" placeholder="Enter f">
35.
36.     <label for="N">Number of Petals (N):</label>
37.     <input type="number" id="N" placeholder="Enter N">
38.
39.     <label for="M">Number of Points on Petal (M):</label>
40.     <input type="number" id="M" placeholder="Enter M">
41.
42.     <label for="f_min">Minimum Frequency (f_min in GHz):</label>
43.     <input type="number" id="f_min" placeholder="Enter f_min">
44.
45.     <label for="f_max">Maximum Frequency (f_max in GHz):</label>
46.     <input type="number" id="f_max" placeholder="Enter f_max">
```

```
47.
48.    <label for="k">Reflector Efficiency (k):</label>
49.    <input type="number" id="k" step="0.01" placeholder="Enter efficiency (k)" value="0.6">
50.
51.    <!-- Select units -->
52.    <label for="units">Select Units:</label>
53.    <select id="units">
54.        <option value="mm">Millimeters (mm)</option>
55.        <option value="cm">Centimeters (cm)</option>
56.        <option value="m">Meters (m)</option>
57.    </select>
58.
59.    <!-- Input for calculation precision (eps) -->
60.    <label for="eps">Calculation Precision (eps):</label>
61.    <input type="number" id="eps" placeholder="Enter eps" step="1e-12" value="1.0e-10">
62.
63.    <button id="generateBtn">Generate and Download Files</button>
64.
65.    <!-- Div for displaying output -->
66.    <div id="output">The results will appear here...</div>
67.
68.    <script>
69.        document.getElementById('generateBtn').addEventListener('click', function () {
70.            try {
71.                console.clear();
72.                console.log("Button clicked, starting process...");
73.
74.                // Clear old output
75.                document.getElementById('output').innerHTML = "Processing...";
76.
77.                // Get input values
78.                const R = parseFloat(document.getElementById('R').value);
79.                const f = parseFloat(document.getElementById('f').value);
80.                const N = parseInt(document.getElementById('N').value);
81.                const M = parseInt(document.getElementById('M').value);
82.                const f_min = parseFloat(document.getElementById('f_min').value) * 1e9; // GHz to Hz
83.                const f_max = parseFloat(document.getElementById('f_max').value) * 1e9; // GHz to Hz
84.                const k = parseFloat(document.getElementById('k').value);
85.                const units = document.getElementById('units').value;
86.                const eps = parseFloat(document.getElementById('eps').value); // Precision for calculation
87.
88.                console.log("Input values received...");
89.
90.                // Validate input
91.                if (isNaN(R) || isNaN(f) || isNaN(N) || isNaN(M) || isNaN(f_min) || isNaN(f_max) ||
92.                    isNaN(k) || isNaN(eps)) {
93.                        alert("Please enter valid values for all fields.");
94.                        return;
95.                    }
96.
97.                    // Basic calculations (following Python code closely)
98.                    const pi = Math.PI;
99.                    const c = 2.99792458e8; // Speed of light in m/s
100.                   const lambda_max = c / f_min;
101.                   const lambda_min = c / f_max;
102.
103.                   let radius;
104.                   if (units === 'mm') {
105.                       radius = R / 1000.0;
106.                   } else if (units === 'cm') {
107.                       radius = R / 100.0;
108.                   } else {
109.                       radius = R;
110.                   }
111.
112.                   const Gain_min = 10.0 * Math.log10(k * (2.0 * pi * radius / lambda_max) ** 2);
113.                   const Gain_max = 10.0 * Math.log10(k * (2.0 * pi * radius / lambda_min) ** 2);
114.
115.                   // Parabolic coefficient calculation
116.                   const a = 1.0 / (4.0 * f);
117.                   const theta = 2.0 * pi / N;
118.                   const value = Math.sqrt(1.0 + 4.0 * a ** 2 * R ** 2);
119.               }
120.           }
121.       }
122.   
```

```

118.     const L = (R * value / 2.0) + Math.log(2.0 * a * R + value) * f;
119.
120.     // Calculating radius r using method of "dividing by half"
121.     function rl(length) {
122.         let left = 0.0;
123.         let right = R;
124.         let r = (left + right) / 2.0;
125.         let value = Math.sqrt(1.0 + 4.0 * a ** 2 * r ** 2);
126.         value = r * value / 2.0 + Math.log(2.0 * a * r + value) * f - length;
127.         let i = 1;
128.         while (Math.abs(value) > eps && i < 100) {
129.             if (value < 0.0) {
130.                 left = r;
131.             } else if (value > 0.0) {
132.                 right = r;
133.             }
134.             r = (left + right) / 2.0;
135.             value = Math.sqrt(1.0 + 4.0 * a ** 2 * r ** 2);
136.             value = r * value / 2.0 + Math.log(2.0 * a * r + value) * f - length;
137.             i++;
138.         }
139.         return r;
140.     }
141.
142.     // Initialize arrays
143.     const l = new Array(M).fill(0);
144.     const r = new Array(M).fill(0);
145.     const D = new Array(M).fill(0);
146.     const eff = new Array(M).fill(0);
147.     const q = new Array(M).fill(0);
148.     const s = new Array(M).fill(0);
149.     const w = new Array(M).fill(0);
150.
151.     l[M - 1] = L;
152.     r[M - 1] = R;
153.     D[M - 1] = R * Math.sqrt(1.0 + 4.0 * a ** 2 * R ** 2);
154.     eff[0] = theta / 2.0;
155.     eff[M - 1] = theta / (2.0 * Math.sqrt(1.0 + 4.0 * a ** 2 * R ** 2));
156.     q[M - 1] = D[M - 1] * (1.0 - Math.cos(eff[M - 1]));
157.     s[M - 1] = l[M - 1] - q[M - 1];
158.     w[M - 1] = D[M - 1] * Math.sin(eff[M - 1]);
159.
160.     // Perform the calculations for each point
161.     for (let i = 1; i < M - 1; i++) {
162.         const length = i * L / (M - 1);
163.         l[i] = length;
164.         r[i] = rl(length);
165.         D[i] = r[i] * Math.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2);
166.         eff[i] = theta / (2.0 * Math.sqrt(1.0 + 4.0 * a ** 2 * r[i] ** 2));
167.         q[i] = D[i] * (1.0 - Math.cos(eff[i]));
168.         s[i] = l[i] - q[i];
169.         w[i] = D[i] * Math.sin(eff[i]);
170.     }
171.
172.     // Generate CSV content
173.     let csvContent = "l,r,D,eff,q,s,w\n";
174.     for (let i = 0; i < M; i++) {
175.         csvContent +=
` ${l[i].toFixed(4)}, ${r[i].toFixed(4)}, ${D[i].toFixed(4)}, ${eff[i].toFixed(4)}, ${q[i].toFixed(4)}, ${s[i].toFixed(4)}, ${w[i].toFixed(4)}\n`;
176.     }
177.
178.     // Create and download CSV file
179.     const blobCsv = new Blob([csvContent], { type: "text/csv" });
180.     const linkCsv = document.createElement("a");
181.     linkCsv.href = URL.createObjectURL(blobCsv);
182.     linkCsv.download = "Parabolic_profile_data.csv";
183.     document.body.appendChild(linkCsv);
184.     linkCsv.click();
185.     document.body.removeChild(linkCsv);
186.
187.     console.log("CSV file generated.");

```

```

188.
189.          // Generate text file with design parameters
190.          let designParameters = '';
191.          designParameters += `Dish radius = ${R} ${units}\n`;
192.          designParameters += `Dish height = ${((a * R ** 2).toFixed(4))} ${units}\n`;
193.          designParameters += `Focus length of the reflector = ${f} ${units}\n`;
194.          designParameters += `Petal length = ${L.toFixed(4)} ${units}\n`;
195.          designParameters += `Number of petals = ${N}\n`;
196.          designParameters += `Petal radius of curvature = ${D[M - 1].toFixed(4)} ${units}\n`;
197.          designParameters += `Minimum frequency = ${((f_min / 1.0e9).toFixed(4))} GHz\n`;
198.          designParameters += `Maximum frequency = ${((f_max / 1.0e9).toFixed(4))} GHz\n`;
199.          designParameters += `Minimum wavelength = ${lambda_min.toFixed(4)} m\n`;
200.          designParameters += `Maximum wavelength = ${lambda_max.toFixed(4)} m\n`;
201.          designParameters += `Reflector efficiency = ${k}\n`;
202.          designParameters += `Minimum antenna gain = ${Gain_min.toFixed(4)} dB\n`;
203.          designParameters += `Maximum antenna gain = ${Gain_max.toFixed(4)} dB\n`;
204.
205.          // Create and download text file
206.          const blobTxt = new Blob([designParameters], { type: "text/plain" });
207.          const linkTxt = document.createElement("a");
208.          linkTxt.href = URL.createObjectURL(blobTxt);
209.          linkTxt.download = "Design_parameters.txt";
210.          document.body.appendChild(linkTxt);
211.          linkTxt.click();
212.          document.body.removeChild(linkTxt);
213.
214.          console.log("Text file with design parameters generated.");
215.
216.          // Output result
217.          document.getElementById('output').innerHTML = "<h3>CSV and Text files generated and
downloaded.</h3>";
218.
219.      } catch (error) {
220.          console.error("Error encountered:", error);
221.          alert("An error occurred. Check the console for more information.");
222.      }
223.  });
224. </script>
225. </body>
226. </html>

```

## Spherical Petal Design

Sphere Radius (R):

Number of Petals (N):

Number of Points on Petal (M):

## Sphere.html

```

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.     <meta charset="UTF-8">
5.     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.     <title>Spherical Petal Design</title>
7.     <style>
8.         body {
9.             font-family: Arial, sans-serif;
10.            margin: 20px;
11.        }
12.        label, input {
13.            margin: 5px 0;
14.            display: block;
15.        }

```

```
16.         button {
17.             margin: 10px 0;
18.         }
19.     </style>
20. </head>
21. <body>
22.     <h1>Spherical Petal Design</h1>
23.
24.     <label for="R">Dish Radius (R):</label>
25.     <input type="number" id="R" placeholder="Enter Dish Radius (R)" step="0.1">
26.
27.     <label for="N">Number of Petals (N):</label>
28.     <input type="number" id="N" placeholder="Enter Number of Petals (N)" step="1">
29.
30.     <label for="M">Number of Points on Petal (M):</label>
31.     <input type="number" id="M" placeholder="Enter Number of Points (M)" step="1">
32.
33.     <button id="downloadBtn">Generate and Download CSV</button>
34.
35.     <script>
36.         document.getElementById('downloadBtn').addEventListener('click', function() {
37.             // Get input values
38.             const R = parseFloat(document.getElementById('R').value);
39.             const N = parseInt(document.getElementById('N').value);
40.             const M = parseInt(document.getElementById('M').value);
41.
42.             // Check if any of the inputs are empty or invalid
43.             if (isNaN(R) || isNaN(N) || isNaN(M) || R <= 0 || N <= 0 || M <= 0) {
44.                 alert('Please enter valid positive values for R, N, and M.');
45.                 return; // Do nothing if values are invalid
46.             }
47.
48.             // Calculate angular width and petal length
49.             const pi = Math.PI;
50.             const theta = 2.0 * pi / N; // angular width of the petal
51.             const L = pi * R / 2.0; // petal length
52.
53.             // Initialize arrays
54.             const l = new Array(M).fill(0);
55.             const r = new Array(M).fill(0);
56.             const eff = new Array(M).fill(0);
57.             const q = new Array(M).fill(0);
58.             const s = new Array(M).fill(0);
59.             const w = new Array(M).fill(0);
60.
61.             // Set the last points
62.             l[M - 1] = L;
63.             r[M - 1] = R;
64.             s[M - 1] = L;
65.             w[M - 1] = theta * R / 2.0;
66.
67.             // Loop through the points and calculate the values
68.             for (let i = 0; i < M - 1; i++) {
69.                 const length = i * L / (M - 1);
70.                 l[i] = length;
71.                 r[i] = R * Math.sin(length / R);
72.                 const sqroot = Math.sqrt(R ** 2 - r[i] ** 2);
73.                 eff[i] = theta * sqroot / (2.0 * R);
74.                 q[i] = r[i] * (1.0 - Math.cos(eff[i])) - eff[i] ** 2 / 2.0) / sqroot +
75.                     r[i] * theta ** 2 * sqroot / (8.0 * R);
76.                 s[i] = l[i] - q[i];
77.                 w[i] = r[i] * (Math.sin(eff[i]) - eff[i]) / sqroot + r[i] * theta / 2.0;
78.             }
79.
80.             // Prepare the CSV data
81.             let csvContent = 'l, r, eff, q, s, w\n';
82.             for (let i = 0; i < M; i++) {
83.                 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`;
84.             }
85.
86.             // Create a Blob from the CSV data
```

```
87.     const blob = new Blob([csvContent], { type: 'text/csv' });
88.     const url = URL.createObjectURL(blob);
89.
90.     // Create a temporary link and trigger download
91.     const a = document.createElement('a');
92.     a.href = url;
93.     a.download = 'Spherical_profile_data.csv';
94.     document.body.appendChild(a);
95.     a.click();
96.     document.body.removeChild(a);
97.     URL.revokeObjectURL(url); // Free up memory
98.   });
99. </script>
100. </body>
101. </html>
```

## Создание электронных файлов для лазерной резки лепестков

На данном этапе проекта мы не использовали лазерную резку, поэтому не имеем соответствующего опыта. Однако мы решили включить этот раздел, чтобы продемонстрировать, каким образом это будет осуществляться на следующих этапах.

Диаметр лазерного луча в резаках обычно варьируется в зависимости от типа лазера и системы фокусировки. Вот несколько ориентировочных значений для распространенных типов лазеров:

### CO2 лазеры

- Диаметр луча обычно составляет 0.1 мм до 0.3 мм.
- Это один из наиболее часто используемых типов лазеров для резки тонких материалов, таких как фанера толщиной 3 – 6 мм.

### Оптоволоконные лазеры

- Диаметр луча обычно меньше, чем у CO2 лазеров, и может составлять 0.05 мм до 0.1 мм.
- Такие лазеры используются реже для фанеры, но обеспечивают более точную резку при необходимости.

### Диодные лазеры

- Диаметр луча может составлять 0.1 мм до 0.5 мм, в зависимости от фокусировки.
- Это более доступные варианты лазеров для небольших домашних станков, но их мощность меньше, чем у CO2 и оптоволоконных.

Таким образом, для тонкой фанеры чаще всего используется CO2 лазер с диаметром луча около 0.1 – 0.3 мм. Выбор диаметра может зависеть от требований к точности резки и параметров настройки резака.

Лазером будут вырезаться как лепестки рефлектора и сферы, так и поддерживающие их рамы. Мы обсуждаем пока только вырезание лепестков, т.к. рамы еще не были разработаны. После расчета профиля лепестка необходимо создать специальный электронный файл для лазерной резки. Лазерные резаки используют файлы форматов, которые поддерживаются системами ЧПУ, например, DXF, G-code, и SVG.

Рассмотрим сначала резку сферического лепестка, который проще. Будет создан алгоритм на JavaScript, который подготовит G-код для резки лепестка и сохранит его в файле соответствующего формата. Алгоритм работает в следующей последовательности:

#### Чтение профиля из CSV-файла

Данные рассчитанного профиля были сохранены в Sphere\_profile\_data.csv в колонках 5 и 6:  $s_i$  (X-координата) и  $w_i$  (Y-координата). Эти координатычитываются и интерпретируются как точки на траектории движения лазерного луча. Для резки лазером, нужно создать профиль с большим числом точек (M), чтобы как можно точнее воспроизвести форму. Поскольку резка неручная, то можно выбрать сотни точек.

#### Компенсация диаметра лазера

Для компенсации ширины прожигаемого шва, к Y-координатам лепестка будет прибавляться половина диаметра лазерного луча.

#### Создание полного профиля

Поскольку вычисленный профиль в Sphere\_profile\_data.csv симметричен относительно оси X, он будет зеркально отображен для создания второй половины лепестка. Получающиеся две ветви профиля исходят из одной точки. Их свободные концы необходимо соединить прямым отрезком (лепесток для сферы). В результате получится замкнутый контур лепестка, готовый к вырезанию. Для перемещения лазерного луча по прямой линии, соединяющей две точки, дополнительные точки на прямой не понадобятся, что упрощает код.

#### Генерация G-кода

Для каждой точки профиля генерируется команда перемещения лазера (команда G1) с указанием точных координат X и Y. Команды включения и выключения лазера (G1 Z0 для начала резки и G0 Z5 для поднятия лазера) добавляются, чтобы правильно управлять процессом резки.

Для создания G-файла мы сгенерили JavaScript-код сохраненный в [репозитории проекта](#) и показанный ниже вместе с интерфейсом в браузере. Для проверки формы лепестка в окне интерфейса необходимо кликнуть на «Generate G-code and Visualize Profile». Если все правильно, кликните на «Download G-code», чтобы сохранить G-файл Full\_sphere\_profile\_gcode.nc.

Upload "Spherical\_profile\_data.csv" with half-profile data:  
 Spherical\_profile\_data.csv

Enter laser beam diameter (same units as profile):

Enter cutting speed (mm/min):

Set laser power (%):

Set laser lift height (mm):

Set laser start delay (ms):

Set laser end delay (ms):

Enter number of passes:

**Profile Visualization:**

[Download G-code](#)

Пример настроек для GUI:

- Скорость резки — Поле ввода числового значения в мм/мин.
- Мощность лазера — Ползунок для регулировки от 0% до 100%.
- Высота подъема лазера — Поле для ввода значений высоты (например, 5 мм).
- Задержка включения/выключения лазера — Поля для задержки в миллисекундах перед началом резки и после завершения (например, 100 мс).
- Количество проходов — Поле для ввода количества проходов (например, 1, 2, 3...).

## Cutting\_sphere\_petal.html

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.     <meta charset="UTF-8">
5.     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.     <title>Laser Cutting Profile Generator</title>
7.     <style>
8.         body {
9.             font-family: Arial, sans-serif;
10.            margin: 20px;
11.        }
12.        label, input, button {
13.            display: block;
14.            margin: 10px 0;
15.        }
16.        canvas {
17.            border: 1px solid black;
18.        }
19.    </style>
20. </head>
21. <body>
22.
23.     <h1>Generate G-code for Laser Cutting with Profile Visualization</h1>
24.
25.     <label for="csvFile">Upload "Spherical_profile_data.csv" with half-profile data:</label>
26.     <input type="file" id="csvFile" accept=".csv">
27.
28.     <label for="laserDiameter">Enter laser beam diameter (same units as profile):</label>
29.     <input type="number" id="laserDiameter" placeholder="Enter beam diameter">
30.
31.     <label for="feedRate">Enter cutting speed (mm/min):</label>
32.     <input type="number" id="feedRate" placeholder="Enter speed (default: 1000)" value="1000">
33.
34.     <label for="laserPower">Set laser power (%):</label>
35.     <input type="range" id="laserPower" min="0" max="100" value="100">
36.
37.     <label for="laserHeight">Set laser lift height (mm):</label>
38.     <input type="number" id="laserHeight" placeholder="Enter height (default: 5 mm)" value="5">
39.
40.     <label for="laserDelayStart">Set laser start delay (ms):</label>
41.     <input type="number" id="laserDelayStart" placeholder="Enter delay (default: 0 ms)" value="0">
42.
43.     <label for="laserDelayEnd">Set laser end delay (ms):</label>
44.     <input type="number" id="laserDelayEnd" placeholder="Enter delay (default: 0 ms)" value="0">
45.
46.     <label for="laserPasses">Enter number of passes:</label>
47.     <input type="number" id="laserPasses" placeholder="Enter number of passes (default: 1)" value="1">
48.
49.     <button id="generateGCode">Generate G-code and Visualize Profile</button>
50.
51.     <h3>Profile Visualization:</h3>
52.     <canvas id="profileCanvas" width="500" height="500"></canvas>
53.
54.     <!-- Placeholder for download link -->
55.     <div id="downloadLink"></div>
56.
57.     <script>
58.         document.getElementById('generateGCode').addEventListener('click', function() {
59.             const csvFile = document.getElementById('csvFile').files[0];
60.             const laserDiameter = parseFloat(document.getElementById('laserDiameter').value);
61.             const feedRate = parseFloat(document.getElementById('feedRate').value);
62.             const laserPower = parseFloat(document.getElementById('laserPower').value);
63.             const laserHeight = parseFloat(document.getElementById('laserHeight').value);
64.             const laserDelayStart = parseFloat(document.getElementById('laserDelayStart').value);
65.             const laserDelayEnd = parseFloat(document.getElementById('laserDelayEnd').value);
66.             const laserPasses = parseInt(document.getElementById('laserPasses').value);
67.
68.             if (!csvFile) {
69.                 alert('Please upload the "Spherical_profile_data.csv" file.');
70.                 return;
71.             }
72.             // Generate G-code logic here
73.             // Visualize profile logic here
74.         });
75.     </script>
76.
```

```
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');
```

```

141.         // Clear the canvas
142.         ctx.clearRect(0, 0, canvas.width, canvas.height);
143.
144.         // Scale the profile to fit into the canvas (maintaining aspect ratio)
145.         const scale = Math.min(canvas.width / rangeX, canvas.height / rangeY) * 0.95; // Adjust
146.         scale to fill canvas
147.
148.
149.
150.
151.         at the first point
152.
153.         Draw each segment
154.
155.
156.
157.
158.         // Generate G-code
159.         let gcode = "G21 ; Set units to mm\n";
160.         gcode += `G90 ; Absolute positioning\n`;
161.         gcode += `S${laserPower} ; Set laser power to ${laserPower}%\n`;
162.
163.         for (let pass = 0; pass < laserPasses; pass++) {
164.             gcode += `G0 Z${laserHeight} ; Raise laser to ${laserHeight} mm\n`;
165.             gcode += `G4 P${laserDelayStart} ; Start delay ${laserDelayStart} ms\n`;
166.
167.             gcode += `G0 X${fullXCoords[0].toFixed(4)} Y${fullYCoords[0].toFixed(4)} ; Move to
starting point\n`;
168.             gcode += "G1 Z0 ; Lower laser for cutting\n";
169.
170.             for (let i = 0; i < fullXCoords.length; i++) {
171.                 gcode += `G1 X${fullXCoords[i].toFixed(4)} Y${fullYCoords[i].toFixed(4)}
F${feedRate} ; Cutting to (${fullXCoords[i].toFixed(4)}, ${fullYCoords[i].toFixed(4)})\n`;
172.             }
173.
174.             gcode += `G4 P${laserDelayEnd} ; End delay ${laserDelayEnd} ms\n`;
175.             gcode += "G0 Z5 ; Raise the laser\n"; // Finish pass
176.         }
177.
178.         gcode += "M05 ; Turn off the laser\n";
179.         gcode += "M30 ; End of program\n";
180.
181.         // Create a download link for the G-code file
182.         const blob = new Blob([gcode], { type: "text/plain" });
183.         const link = document.createElement('a');
184.         link.href = URL.createObjectURL(blob);
185.         link.download = 'Full_sphere_profile_gcode.nc';
186.         link.innerText = 'Download G-code';
187.
188.         // Replace existing link (if any)
189.         const downloadLinkDiv = document.getElementById('downloadLink');
190.         downloadLinkDiv.innerHTML = ''; // Clear previous link
191.         downloadLinkDiv.appendChild(link);
192.     };
193.
194.     reader.readAsText(csvFile);
195.   });
196. 
```

Создание замкнутого профиля лепестка для параболоида немного сложнее, т.к. свободные концы ветвей профиля соединяются дугой, а не прямым отрезком, как в случае сферы. Вспомним структуру файла Parabolic\_profile\_data.csv, показанную на Рис. 4. Считая, что кривые профиля выходят из точки с координатой  $X = 0$  и  $Y = 0$ , алгоритм на JavaScript для координат точек замыкающей дуги будет следующим:

```
// 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);
}
```

где  $l_{max}$  – длина лепестка (последнее значение в  $l$ -колонке файла Parabolic\_profile\_data.csv),  $effmax$  – максимальный угол лепестка (последнее значение в  $eff$ -колонке), и  $D_{max}$  – максимальная длина касательного отрезка (последнее значение в  $D$ -колонке). Причем величина  $D_{max}$  уже была увеличена на половину диаметра лазерного луча, хотя такая точность едва ли понадобится при сборке рефлектора. Координаты центра дуги (см. Рис. 6) будут  $X = (l_{max} - D_{max})$  и  $Y = 0$ .

Программный код на JavaScript вместе с интерфейсом в браузере показаны ниже. Его можно скачать из [репозитория проекта](#). Для проверки формы лепестка в окне интерфейса необходимо кликнуть на «Generate G-code and Visualize Profile». Если все правильно, кликните на «Download G-code», чтобы сохранить G-файл Full\_paraboloid\_profile\_gcode.nc.

## Generate Parabolic Profile with Arc Closure

Upload "Spherical\_profile\_data.csv" with half-profile data:

Parabolic\_profile\_data.csv

Enter laser beam diameter (same units as profile):

Enter number of points on the arc (M):

Enter cutting speed (mm/min):

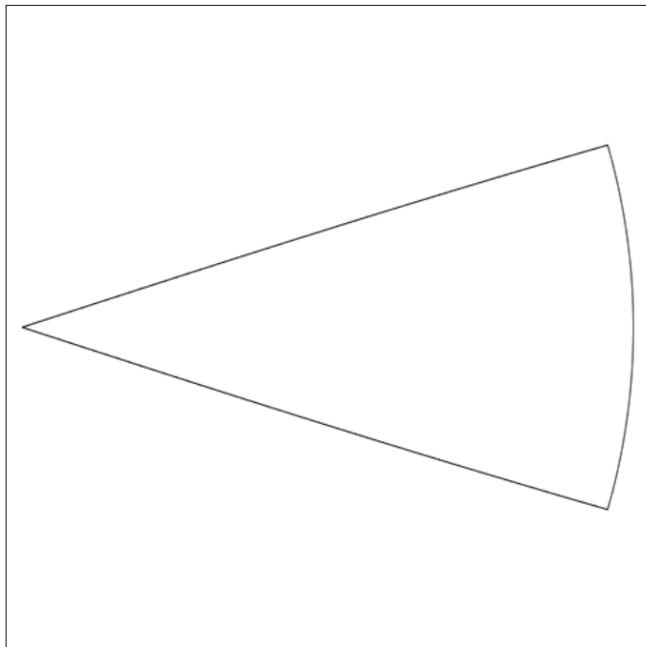
Set laser power (%):

Set laser lift height (mm):

Set laser start delay (ms):

Set laser end delay (ms):

Enter number of passes:

**Profile Visualization:**[Download G-code](#)**Cutting\_paraboloid\_petal.html**

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.   <meta charset="UTF-8">
5.   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.   <title>Laser Cutting Profile with Arc Closure</title>
7.   <style>
8.     body {
9.       font-family: Arial, sans-serif;
10.      margin: 20px;
11.    }
12.    label, input, button {
13.      display: block;
14.      margin: 10px 0;
15.    }
16.    canvas {
17.      border: 1px solid black;
18.    }
19.  </style>
20. </head>
21. <body>
22.
23.   <h1>Generate Parabolic Profile with Arc Closure</h1>
24.
25.   <label for="csvFile">Upload "Spherical_profile_data.csv" with half-profile data:</label>
26.   <input type="file" id="csvFile" accept=".csv">
27.
28.   <label for="laserDiameter">Enter laser beam diameter (same units as profile):</label>
29.   <input type="number" id="laserDiameter" placeholder="Enter beam diameter">
30.
31.   <label for="M">Enter number of points on the arc (M):</label>
32.   <input type="number" id="M" placeholder="Enter number of points on the arc" value="100">
33.
34.   <label for="feedRate">Enter cutting speed (mm/min):</label>
35.   <input type="number" id="feedRate" placeholder="Enter speed (default: 1000)" value="1000">
36.
37.   <label for="laserPower">Set laser power (%):</label>
38.   <input type="range" id="laserPower" min="0" max="100" value="100">
39.
40.   <label for="laserHeight">Set laser lift height (mm):</label>
```

```
41.      <input type="number" id="laserHeight" placeholder="Enter height (default: 5 mm)" value="5">
42.
43.      <label for="laserDelayStart">Set laser start delay (ms):</label>
44.      <input type="number" id="laserDelayStart" placeholder="Enter delay (default: 0 ms)" value="0">
45.
46.      <label for="laserDelayEnd">Set laser end delay (ms):</label>
47.      <input type="number" id="laserDelayEnd" placeholder="Enter delay (default: 0 ms)" value="0">
48.
49.      <label for="laserPasses">Enter number of passes:</label>
50.      <input type="number" id="laserPasses" placeholder="Enter number of passes (default: 1)" value="1">
51.
52.      <button id="generateGCode">Generate G-code and Visualize Profile</button>
53.
54.      <h3>Profile Visualization:</h3>
55.      <canvas id="profileCanvas" width="500" height="500"></canvas>
56.
57.      <!-- Placeholder for download link -->
58.      <div id="downloadLink"></div>
59.
60.      <script>
61.          document.getElementById('generateGCode').addEventListener('click', function() {
62.              const csvFile = document.getElementById('csvFile').files[0];
63.              const laserDiameter = parseFloat(document.getElementById('laserDiameter').value);
64.              const feedRate = parseFloat(document.getElementById('feedRate').value);
65.              const laserPower = parseFloat(document.getElementById('laserPower').value);
66.              const laserHeight = parseFloat(document.getElementById('laserHeight').value);
67.              const laserDelayStart = parseFloat(document.getElementById('laserDelayStart').value);
68.              const laserDelayEnd = parseFloat(document.getElementById('laserDelayEnd').value);
69.              const laserPasses = parseInt(document.getElementById('laserPasses').value);
70.              const M = parseInt(document.getElementById('M').value); // Number of points on the arc
71.
72.              if (!csvFile) {
73.                  alert('Please upload the "Spherical_profile_data.csv" file.');
74.                  return;
75.              }
76.
77.              if (isNaN(laserDiameter) || laserDiameter <= 0) {
78.                  alert('Please enter a valid laser beam diameter.');
79.                  return;
80.              }
81.
82.              if (isNaN(M) || M <= 10) {
83.                  alert('Please enter a valid number of points (M > 10).');
84.                  return;
85.              }
86.
87.              const reader = new FileReader();
88.              reader.onload = function(event) {
89.                  const csvData = event.target.result;
90.                  const lines = csvData.split('\n');
91.                  const xCoords = []; // 6th column (X axis)
92.                  const yCoords = []; // 7th column (Y axis)
93.                  let effmax, Dmax, lmax;
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 >= 7) { // Ensure there are enough columns
99.                          const x = parseFloat(columns[5]); // 6th column (index 5)
100.                         const y = parseFloat(columns[6]); // 7th column (index 6)
101.                         const eff = parseFloat(columns[3]); // 4th column (eff)
102.                         const D = parseFloat(columns[2]); // 3rd column (D)
103.                         const l = parseFloat(columns[0]); // 1st column (l)
104.
105.                         if (!isNaN(x) && !isNaN(y) && !isNaN(eff)) {
106.                             xCoords.push(x);
107.                             yCoords.push(y + laserDiameter / 2); // Compensate laser burn by adding half
diameter
108.                             lmax = l; // Last value in the 1st column
109.                             Dmax = D + laserDiameter / 2; // Last value in the 3rd column
110.                             effmax = eff; // Last value in the 4th column
111.                         }
112.                     }
113.                 }
114.             }
115.         }
116.     }
117. 
```

```
112.         }
113.     }
114.
115.     // Check if data is valid
116.     if (xCoords.length === 0 || yCoords.length === 0) {
117.         alert('No valid data found in the file.');
118.         return;
119.     }
120.
121.     // Create arrays for the arc closure
122.     const arcXCoords = [];
123.     const arcYCoords = [];
124.
125.     // Algorithm to calculate arc points
126.     for (let i = 0; i < M; i++) {
127.         const eff = effmax - (2.0 * effmax * i) / (M - 1);
128.         const arcX = Dmax * Math.cos(eff) + lmax - Dmax;
129.         const arcY = Dmax * Math.sin(eff);
130.         arcXCoords.push(arcX);
131.         arcYCoords.push(arcY);
132.     }
133.
134.     // Create arrays for the full profile (including arc and mirrored points)
135.     const fullXCoords = [...xCoords];
136.     const fullYCoords = [...yCoords];
137.
138.     // Add arc points to close the profile
139.     fullXCoords.push(...arcXCoords);
140.     fullYCoords.push(...arcYCoords);
141.
142.     // Mirror the profile by reflecting relative to X-axis (keep X the same and reverse Y)
143.     for (let i = xCoords.length - 1; i >= 0; i--) {
144.         fullXCoords.push(xCoords[i]); // X-axis stays the same
145.         fullYCoords.push(-yCoords[i]); // Reflect Y relative to the X-axis
146.     }
147.
148.     // Calculate scaling factors to fit the profile into the canvas
149.     const minX = Math.min(...fullXCoords);
150.     const maxX = Math.max(...fullXCoords);
151.     const minY = Math.min(...fullYCoords);
152.     const maxY = Math.max(...fullYCoords);
153.
154.     const rangeX = maxX - minX;
155.     const rangeY = maxY - minY;
156.
157.     const canvas = document.getElementById('profileCanvas');
158.     const ctx = canvas.getContext('2d');
159.
160.     // Clear the canvas
161.     ctx.clearRect(0, 0, canvas.width, canvas.height);
162.
163.     // Scale the profile to fit into the canvas (maintaining aspect ratio)
164.     const scale = Math.min(canvas.width / rangeX, canvas.height / rangeY) * 0.95; // Adjust
scale to fill canvas
165.     const offsetX = canvas.width / 2 - ((maxX + minX) / 2) * scale;
166.     const offsetY = canvas.height / 2 + ((maxY + minY) / 2) * scale;
167.
168.     // Draw the profile on the canvas
169.     ctx.beginPath();
170.     ctx.moveTo(offsetX + fullXCoords[0] * scale, offsetY - fullYCoords[0] * scale); // Start
at the first point
171.     for (let i = 1; i < fullXCoords.length; i++) {
172.         ctx.lineTo(offsetX + fullXCoords[i] * scale, offsetY - fullYCoords[i] * scale); //
}
173.     ctx.closePath(); // Close the path to complete the profile
174.     ctx.stroke(); // Outline the profile
175.
176.     // Generate G-code
177.     let gcode = "G21 ; Set units to mm\n";
178.     gcode += `G90 ; Absolute positioning\n`;
179.     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. 
216. </body>
217. </html>

```

## Заключение

В результате выполнения начального этапа проекта были разработаны и внедрены алгоритмы раскроя лепестков параболоида и сферы, а также алгоритмы их лазерной резки. Определена общая компоновка системы, проработаны ключевые технологические аспекты изготовления рефлектора и защитной сферы. Также был разработан и протестирован метод наведения рефлектора на вышку связи. На следующем этапе проекта планируется дальнейшая доработка, усовершенствование и автоматизация этих методов. Помимо этого, предстоит разработка дополнительных электромеханических устройств для более точного наведения рефлектора и регулирования поляризации облучателя. В дополнение к использованию широкополосных облучателей, доступных на рынке, мы рассматриваем возможность разработки и испытания собственных конструкций.

Вопрос сооружения вышки (Рис. 23) пока остается открытым. Возможен вариант размещения системы на чердаке дома. Однако более высокая установка рефлектора и модема могла бы существенно улучшить качество приема и передачи интернет-сигнала, а также позволила бы расширить зону покрытия Wi-Fi на территорию радиусом до 150–200 метров. Это обеспечило бы возможность подключения к системе нескольких соседей для совместного использования. Кроме того,

расширенное покрытие Wi-Fi обеспечит подключение периферийных устройств, используемых для автоматизации и контроля, таких как теплицы, системы полива, насосные станции, камеры наблюдения и другие.

Недавно появилась [информация](#) о разработке нового гибридного протокола передачи сигнала, объединяющую Wi-Fi и сетевой протокол Long Range (LoRa), названную [WiLo](#). Эта новая концепция беспроводной связи ориентирована на использование существующего оборудования Wi-Fi и LoRa и предназначена для приложений [интернета вещей](#) (IoT), таких как сети сенсоров для сельского хозяйства и умных городов. Дело в том, что Wi-Fi ограничен дальностью и высоким энергопотреблением, тогда как LoRa характеризуется низким потреблением энергии и большим радиусом действия. Технология WiLo объединяет преимущества обеих систем, позволяя использовать стандартные Wi-Fi устройства для передачи данных на большие расстояния без дополнительного оборудования, что снижает стоимость, сложность и потенциальные риски сбоев. Эксперименты показали, что WiLo может успешно работать на расстояниях до 500 метров с эффективностью 96%. Основным преимуществом WiLo является его способность работать на существующем оборудовании без значительных дополнительных затрат. Однако недостатком остается увеличенное энергопотребление Wi-Fi устройств из-за необходимости одновременной обработки связи и эмуляции сигналов. В будущем исследователи планируют повысить энергоэффективность, скорость передачи данных и устойчивость системы к помехам, а также внедрить меры безопасности для кросс-технологической коммуникации.

Использование композитных материалов для создания легких и прочных конструкций представляется весьма перспективным. Внедрение таких технологий в условиях дачных мастерских может стать значимым технологическим достижением.