

Прием и обработка АРТ сигнала со спутников дистанционного зондирования Земли

Тимофеев Андрей
Лебедев Михаил
Капцевич Ольга

25 января 2024 г.

Содержание

Введение	2
APT (Automatic Picture Transmission)	2
RTL-SDR	4
Аппаратная реализация	5
Варианты архитектуры системы	6
Вариант №1	6
Вариант №2	8
Вариант №3	9
Реализация	10
Программная реализация	11
Автоматизация приема АРТ сигнала на RTL-SDR	11
Расчет времени приема спутника	11
Запись и сохранение радио сигнала на нужной волне во время пролета спутника	13
Декодирование принятого АРТ сигнала спутника NOAA-19 . .	14
Передискретизация сигнала до частоты 20800 Гц и Преоб- разование в аналитический сигнал	15
Нормирование значений сигнала от 0 до 255	16
Нахождение положения кадров синхронизации сигнала АРТ	17
Выбор сектора для формирования изображения	18
Список литературы	19

Введение

Суть проекта заключается в создании станции (на основе RTL-SDR) по приему и обработки [APT \(Automatic Picture Transmission\)](#) сигнала со спутников дистанционного зондирования Земли на базе лаборатории спутникового мониторинга ДВФУ.



Рис. 1: Антенны на крыше G корпуса.

APT (Automatic Picture Transmission)

Automatic Picture Transmission - это стандарт для аналоговой передачи данных (изображений) по радиоканалу с использованием специализированных радиопередатчиков, установленных на метеорологических спутниках.

Стандарт АПТ был разработан подразделением [NOAA National Earth Satellite Service](#) в 1960-х. Впервые был описан в статье “[The Automatic](#)

Picture Transmission (APT) TV camera system for meteorological satellites".
Первый спутник на полярной орбите, передававший изображения в формате APT — TIROS-N.

Что касается предываемых данных, то в одном канале непрерывно передаётся изображение с длинноволнового ИК-канала (10,8 микрометров), а в другом канале на освещённой стороне Земли передаётся изображение в ближнем (близком к видимому свету) ИК диапазоне (0,86 микрометра), а на не освещённой — в среднем ИК-диапазоне (3,75 микрометра).

Структура АРТ формата передачи данных:

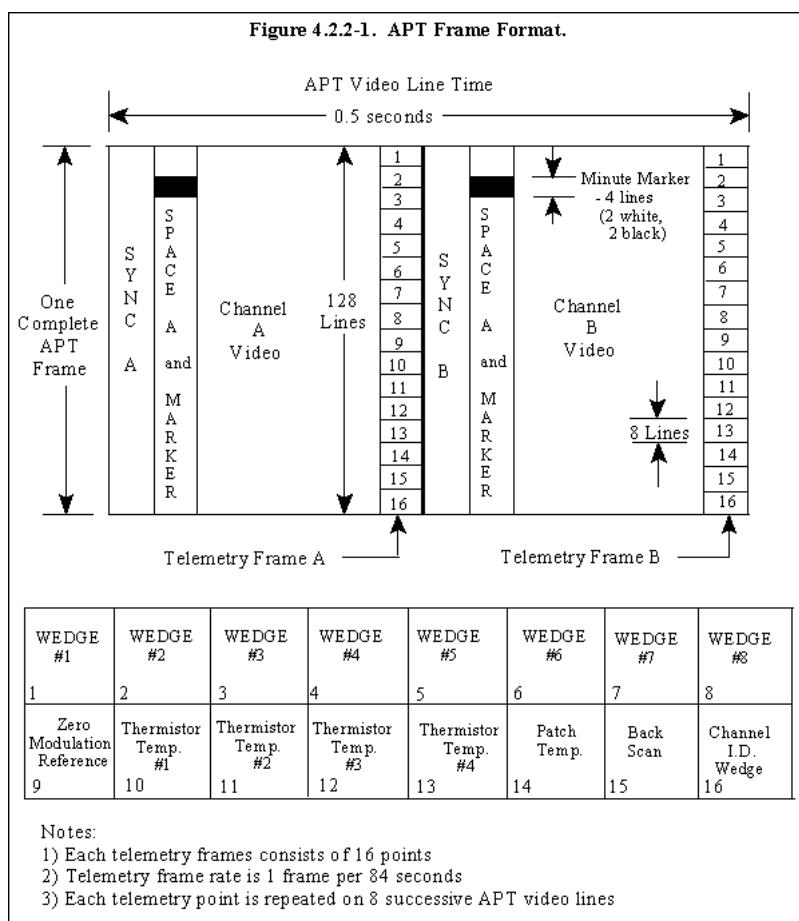


Рис. 2: Структура АРТ.

RTL-SDR

SDR (Software-defined radio) - это радиопередатчик или радиоприёмник, использующий технологию, позволяющую с помощью программного обеспечения устанавливать или изменять рабочие радиочастотные параметры, включая, в частности, диапазон частот, тип модуляции или выходную мощность.



Рис. 3: Пример SDR.

Аппаратная реализация

Для приёма сигнала используются две антенны X-Quad на 400+ МГц и 130+ МГц закреплённые на поворотном устройстве на крыше корпуса G кампуса ДВФУ. Поворотное устройство управляется программно с компьютера, находящегося в кабинете G542 (на крышу протянуты провода).



Рис. 4: Контроллер поворота.

В рамках проекта стояла задача разработать аппаратную систему, позволяющую принимать и обрабатывать радиосигналы с помощью программно определяемых радиосистем (RTL-SDR). Желаемые качества такой системы были следующие:

- **Возможность непрерывной работы.** Предполагается постоянное функционирование системы.
- **Возможность полной автоматизации.** Предполагается использование станции полностью в автоматическом режиме, включая поворот антенны, начало и конец записи.
- **Отказоустойчивость и простота починки.** Система должна быть достаточно простой, а также простой в ремонте при поломке.
- **Возможность дополнения и обновления.** В данный момент используются модули RTL-SDR, но впоследствии желательна возможность заменить их радиосистемами лучшего качества.
- **Низкая шумность.** Желательно избегать длинных линий передачи слабого аналогового сигнала от антенны к радиоприёмнику.

Руководствуясь этими критериями были разработаны 3 возможные архитектуры системы.

Варианты архитектуры системы

В каждом из вариантов архитектуры предполагается расположение части оборудования непосредственно на вращающейся части поворотного комплекса антенн в влагозащищённом контейнере. На схемах ниже эта часть представлена слева и схематически отделена от оборудования в помещении двумя косыми чертами.

Также в каждом из вариантов предполагается питание этого оборудования от источника постоянного тока расположенного в помещении. Для того, чтобы доставить стабильную мощность оборудованию, по линии питания подаётся напряжение 14В и используется регулятор для понижения напряжения до 5В.

Вариант №1

Наиболее предпочтительным вариантом представлялось использование удлинителя USB для подключения RTL-SDR через длинную цифровую линию непосредственно к компьютеру в помещении, на котором развернута программная часть системы.

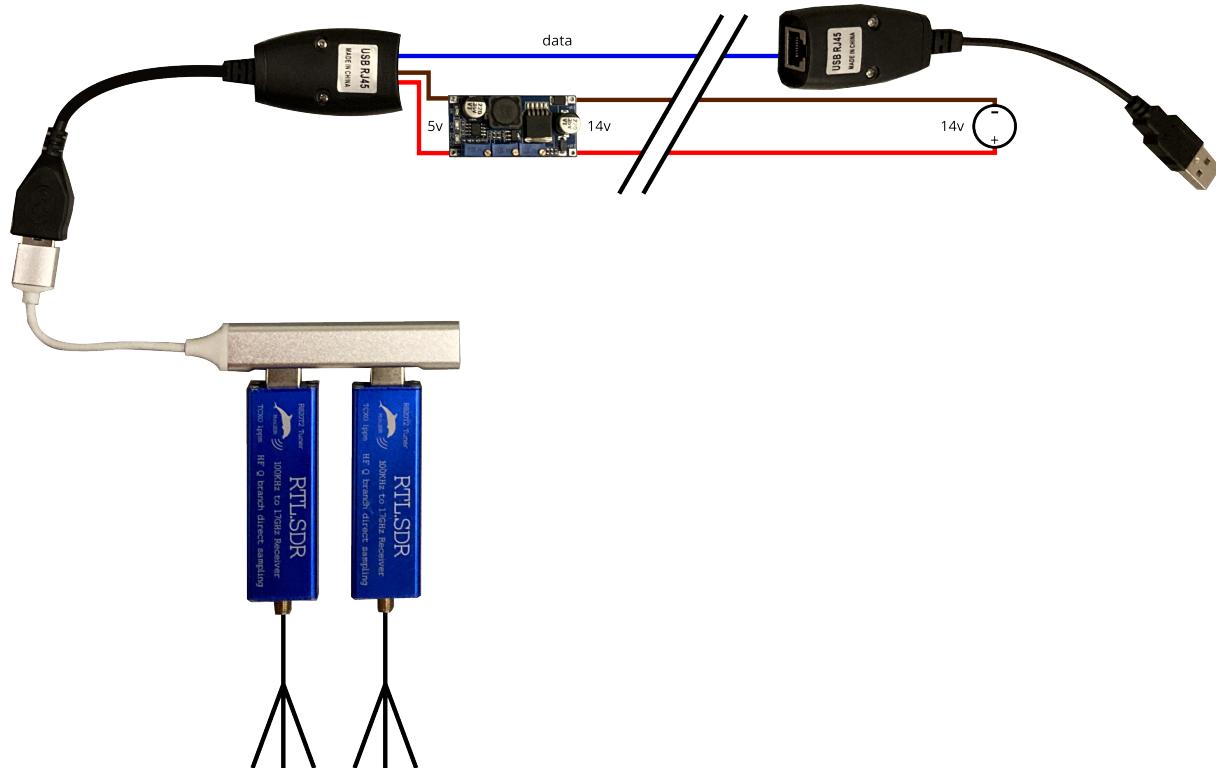


Рис. 5: Вариант архитектуры №1.

Такой вариант позволил бы уменьшить количество оборудования на крыше и его энергопотребление, упростив систему и увеличив отказоустойчивость системы по сравнению в вариантом №2. Однако, требо-

валось проверить, будет ли удлинитель USB предоставлять достаточную скорость передачи данных для стабильной работы системы.

Вариант №2

Другим возможным вариантом представлялось подключение RTL-SDR модулей к микрокомпьютеру Raspberry Pi, который также был бы установлен на крыше, и передачу цифровых данных на управляющий компьютер для декодирования и обработки используя Ethernet.

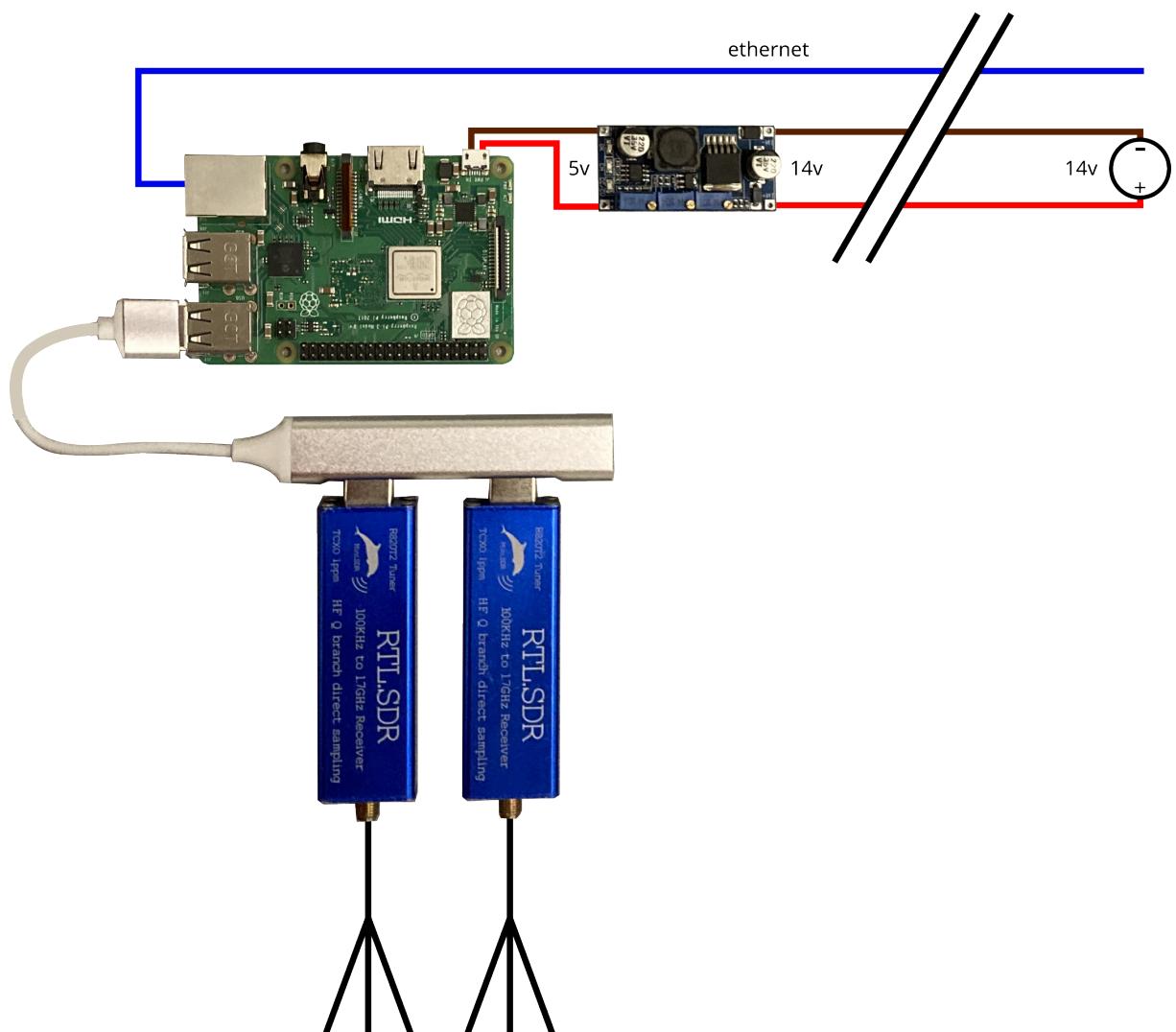


Рис. 6: Вариант архитектуры №2.

Такой вариант обеспечивал бы более высокую пропускную способность и устойчивость к магнитным помехам линии передачи данных. Однако, он требовал бы более высокой мощности питания и дополнительно потребовал бы дополнительного охлаждения, а также усложнил бы программную реализацию, отделив управляющий компьютер и радиоприёмники.

Вариант №3

Последним рассмотренным вариантом являлось расположение на крыше только предусилителей, и передачи аналогового сигнала, от них по коаксиальному проводу к радиоприёмникам, расположенным в помещении.

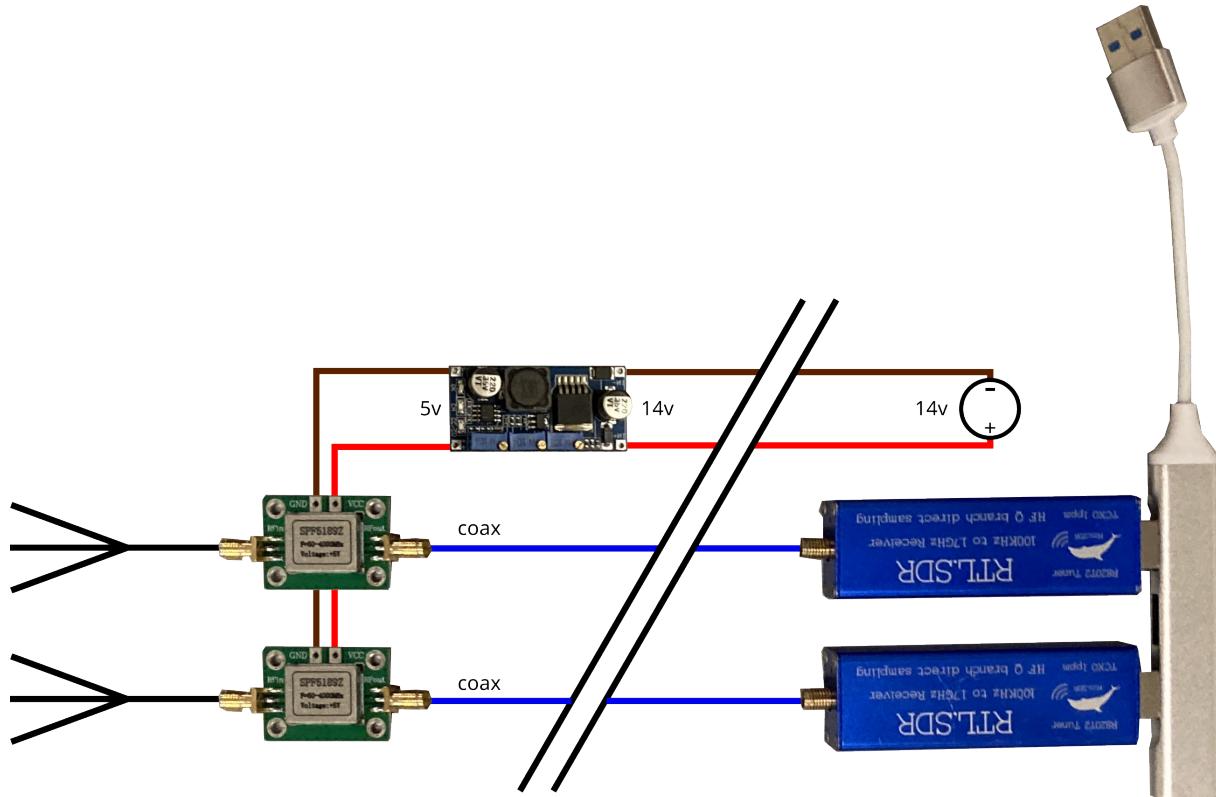


Рис. 7: Вариант архитектуры №3.

Этот вариант был бы наиболее простым в реализации, и обеспечивал бы более простую возможность ремонта и обновления системы. Однако, он был бы наиболее подвержен помехам в виду передачи аналогового а не цифрового сигнала, что дополнительно усугублялось бы возможными наводками по линии питания от инверторного понижающего регулятора.

Реализация

Было проведено тестирование работоспособности вариантов №1 и №2 для необходимой длины линий питания и данных. Было обнаружено, что оба варианты архитектуры обеспечивают достаточное питание, однако удлинителю USB не располагают достаточной шириной канала для работы с RTL-SDR. В силу этого решено было использовать вариант архитектуры №2. Вариант №3 не был протестирован ввиду того, что у нас не было коаксиального кабеля необходимой длины.

В данный момент система собрана, протестирована, и готова к влагоизоляции и установке на крышу, что будет сделано как только нами будет получено соответствующее разрешение от вуза.

Программная реализация

Исходный код скриптов можно найти в следующем [репозитории](#).

Автоматизация приема АРТ сигнала на RTL-SDR

Для автоматизации приема АРТ сигнала был реализован скрипт `script_noaa_data_receiver.py` на языке программирования Python.

Автоматизация заключается в следующем:

- Расчет времени приема спутника.
- Запись и сохранение радио сигнала на нужной волне во время пролета спутника.

Расчет времени приема спутника

Что бы узнать когда мимо нас пролетает конкретный спутник, можно обратиться к уже готовым сервисам предоставляющих эту информацию и забирать её от туда. Таким сервисом стал [N2YO](#).

N2YO - это сервис на котором можно отслеживать положение спутников на орбите в реальном времени, получать о спутниках справочную информацию, а так же предсказывать время пролета спутника в конкретной точке мира, что нам и интересно. Для обмена этой информации у сервиса N2YO есть [API](#).

Для получения этой информации автоматически, была написана функция `get_start_record_time` которая обращается к API сервиса N2YO и возвращает ближайшее время пролета выбранного нами спутника.

```
CONFIG_PATH = 'config.ini'

# N2YO API -> https://www.n2yo.com/api/
def get_start_record_time(satellite_id: str,
                           lat: str,
                           lon: str,
                           days: int
) -> tuple[datetime.datetime, datetime.datetime]:
    """
    Time of passage of the specified satellite in UTC + 10h

    :param satellite_id: satellite id in n2yo
    :param lat: observer's latitude
    :param lon: observer's longitude
    :param days: number of days of prediction (max 10)

    :return: satellite flyby start time and satellite
             flyby end time
    """

```

Полную реализацию функции с описанием, можно найти [тут](#).

Запись и сохранение радио сигнала на нужной волне во время пролета спутника

Запись сигнала осуществлялась на RTL-SDR. Для взаимодействия Python с RTL-SDR использовалась библиотека [SoapySDR](#).

SoapySDR — это набор кроссплатформенных библиотек, написанных на C++, предоставляющих унифицированный доступ к SDR-устройствам. Так же для этой библиотеки существует [оболочка на Python](#), которая и была использована.

Функция *sdr_record* для записи и сохранения радио сигнала на выбранной частоте.

```
def sdr_record(device,
                frequency,
                sample_rate,
                gain,
                blocks_count
) -> None:
    """
    Receiving and recording a radio signal

    :param device: sdr device object
    :param frequency: value of frequency
    :param sample_rate: value of sample rate
    :param gain: value of gain
    :param blocks_count: number of record blocks
    """
```

Полную реализацию функции с описанием, можно найти [тут](#).

Декодирование принятого АРТ сигнала спутника NOAA-19

Для превращения сигнала от спутника NOAA-19 в изображение реализован класс *NOAADecoder* в скрипте *script_noaa_decoder.py* на языке Python.

```
class NOAADecoder:  
    def __init__(self,  
                 black_point: int,  
                 white_point: int,  
                 components: Optional[List[str]])  
        ) -> None:  
    ...  
  
    Class for decode NOAA APT data  
  
    :param black_point: dynamic range lower bound,  
                       percent  
    :param white_point: dynamic range upper bound,  
                       percent  
    :param components: portions of the image to  
                      preserve/filter  
    ...
```

Полную реализацию функции с описанием, можно найти [тут](#).

Декодирование сигнала в изображение происходит следующим образом:

- Передискретизация сигнала до частоты 20800 Гц.
- Преобразование в аналитический сигнал ([Преобразование Гилберта](#)).
- Нормирование значений сигнала от 0 до 255.
- Нахождение положения кадров синхронизации сигнала АРТ.
- Выбор сектора для формирования изображения.

Передискретизация сигнала до частоты 20800 Гц и Преобразование в аналитический сигнал

Сам процесс докодирования сигнала заключается в этих двух последовательных этапах, где на выходе мы имеем аналитический сигнал, где каждое значение представляет собой пиксел, телеметрию или мусорные значения.

За эту часть отвечает функция *audio_to_hilbert*:

```
def audio_to_hilbert(file_path) -> np.ndarray:  
    ...  
    Load the audio and convert to Hilbert transformed  
    amplitude info  
  
    :param file_path: path to wav file  
    :return: amplitude information corresponding to  
            pixel intensity  
    ...
```

Полную реализацию функции с описанием, можно найти [тут](#).

Нормирование значений сигнала от 0 до 255

Из за того, что сигналы могут иметь большой разброс значений, мы нормируем полученные данные, что бы изображение корректно отобразилось.

За это отвечает функция *quantize*:

```
def quantize(data: np.ndarray,
             black_point: int,
             white_point: int) -> np.ndarray:
    ...
    Digitize signal to valid pixel intensities in the
    uint8 range

    :param black_point: dynamic range lower bound,
                       percent
    :param white_point: dynamic range upper bound,
                       percent
    :return: quantized numpy array to uint8
    ...
```

Полную реализацию функции с описанием, можно найти [тут](#).

Нахождение положения кадров синхронизации сигнала АРТ

Что бы корректно определить полосы сканирования и преобразовать одномерный массив данных в двумерное изображение, требуется сопоставить строки по синхросерии, которая известна.

За это отвечает функция *reshape*:

```
SYNCHRONIZATION_SEQUENCE = np.array([0, 0, 255, 255, 0, 0,
                                      255, 255, 0, 0, 255,
                                      255, 0, 0, 255, 255,
                                      0, 0, 255, 255, 0, 0,
                                      255, 255, 0, 0, 255,
                                      255, 0, 0, 0, 0, 0, 0,
                                      0, 0]) - 128

def reshape(data: np.ndarray,
            synchronization_sequence: np.ndarray =
            SYNCHRONIZATION_SEQUENCE,
            minimum_row_separation: int = 2000
) -> np.ndarray:
    """
    Reshape the numpy array to a 2D image array

    :param synchronization_sequence: sequence to
        indicate row start
    :param minimum_row_separation: minimum columns of
        separation
        (a hair less than 2080)
    :return: a 2D reshaped image array
    """

```

Полную реализацию функции с описанием, можно найти [тут](#).

Выбор сектора для формирования изображения

Из полученного двумерного массива данных, нас интересует только *channelA* и *channelB*, так как только они содержат изображения.

За это отвечает функция *select_image_components*:

```
def select_image_components(data: np.ndarray,
                            components:
                            Optional[List[str]] =
                            ['image_a', 'image_b'])
    -> np.ndarray:
        ...
        Select the image components to include

        :param components: portions of the image to
                           preserve/filter
        :return: image array with just the appropriate
                 image components
        ...
```

Полную реализацию функции с описанием, можно найти [тут](#).

Список литературы

1. Как работает декодер АРТ формата
2. SoapySDR for Python
3. Automatic picture transmission