27.07.2016 Начата реализация эмулятора РЛМ старого и нового типа. В связи с сложностью доработки программы во время эксперимента и невозможности нормальной работы станции не во время эксперимента, для устранения ошибок реализации программы было принято решение создание эмулятора работы обоих РЛМ. Это снимет необходимость проведения эксперимента для доработки программ.

20.07.2016 Подготовлена предварительная схема обработки данных с новой РЛС. Согласно протоколу он реализован в виде TCP клиента. Возникли проблемы совместимости системы обработки, так как новый протокол не определяет жестко фиксированный размер пакета. Его размер формируется из количества выборок АЦП и количества импульсов в пачке. Зависимость определяется по формуле Размер = 108 (заголовок) + Количество выборок АЦП \* количество импульсов в пачке \* 1.5 \* 2.

В данной формуле 2 - количество каналов. Сначала подаются данные с одного канала, потом из другого. Подробности протокола в Приложении.

15.07.2016 Создано из rip3p отдельное приложение (rip3pc) с рядом изменений, касающихся обработки данных и его вывода на все графики:

1. Принимаем не жестко зафиксированные пачки, а указанные в соотвествующих полях настройки.
2. Берем дополнительные блоки, для получения еще нескольких разностей фаз.
3. Вместо прибавления к массиву точек результатов сканирования (сырых либо из файлов) мы берем только несколько блоков (количество определено в настройках в поле leRasterPeriod) и выводим точки в соотвествии с разностью фаз.
4. Полученные данные показывают положение объектов по углу места и по высоте.

06.06.2016 - Выставлен таймер синхронизации конфигурационных параметров из класса-хранилища Memory в файл настроек. Этим исправлена высокая нагрузка при изменении параметров графиков. Например при изменении масштаба за секунду значение меняется сотни раз, хотя необходимо только последнее значение. При этом создается большая нагрузка синхронизации потоков, так как всем потокам подается команда, что данные изменились.

03.06.2016 - Для экспериментальной проверке обработки данных от РЛС, получаемый от него массив данных предварительно сохраняется в файл.

1.06.2016 - доработан класс nomain/UDPSock, отвечающий за связь с РЛС. В класс Worker добавлен слот sendMsg для отправки готовых команд на РЛС. В классе MControl добавлен сигнал sendMsgSignal и связан со слотом Worker::sendMsg.

27.05.2016 - В классе formSetting добавлен сигнал sendParam для отправки параметров настройки в РЛС. Он через MControl и Worker связан с nomain/UDPSock. В UDPSock реализована формирование пакета настройки РЛС и передача в SendCmdPacket.

23.05.2016 - 26.05.2016 - Тестирования модуля MathGL, как замена самодельного класса PlotGl. Полученный результат показывает очень низкую скорость динамической отрисовки объектов в данной библиотеке. Принято решение вернуться к PlotGl. Достаточно хорошо получаются статичные графики, но даже если просто изменить координаты или масштаб, сразу уменьшается отклик системы. Вывод: недостаточно оптимально для решаемых задач.

20.05.2016 - в классе nomain/UDPSock реализован метод SendCmdPacket, в котором из отправляемых данных формируется пакет (добавляются дополнительные параметры в начало тела пакета). Получается из глобального класса - хранилища переменных Memory адрес и порт сервера и передается, через метод sendMsg в РЛС.

18.05.2016 - В класс formSetting добавлена форма для изменения адреса и порта РЛС. Форма синхронизирована с глобальным классом - хранилищем переменных Memory.

16.05.2016 - Создан класс UDPSock для обмена командами и данными с РЛС. Класс выделен в отдельный поток, для того, чтобы обработка данных с РЛС не прерывала процесс получения этих данных. Реализованы слоты отправки данных и получения данных.

12.05.2016 - Окончание процесса переноса обработки вторичных данных для вывода на графики в отдельные потоки для улучшения отклика программы.

28.04.2016 - В связи с увеличением количества графиков, принять решение перенести обработку вторичных данных для вывода на графики в отдельные потоки, связанные только с этими графиками. Таким образом будет уменьшена нагрузка на основной поток, соответственно улучшен отклик программы и соответственно появляется возможность наращивания количества графиков. Слабым местом можно считать только невозможность увеличения количества ядер на ПК. В случае когда количество потоков будет превышать количество ядер ОС будет создавать очередь и упадет скорость отображения данных на графиках. В связи с этим необходимо увеличение приоритета потоку обмена данными с РЛС.

22.04.2016 - Реализована система сохранения данных во внутренний формат .3d. Структурно файл состоит из 2-х частей:

1. Амплитудная характеристика сигнала. По умолчанию упорядоченна в блоки по 1024.
2. Фазовая характеристика сигнала. По умолчанию упорядоченна в блоки по 1024.

Размер файла зависит от дальности (по умолчанию 1024) и времени сканирования. Для случаев, когда дальность меньше при загрузке файла появляется окно выбора дальности и времени сканирования. Достаточно ввести один параметр. Второй будет рассчитан относительно размера файла.

Данных хранятся во float, поэтому легко формируется в Mathcad.

18.04.2016 Реализован класс графика gr/PlotOsc - осциллограммы и класс графика амплитудной и фазовой характеристики gr/PlotRaster, в котором, в интерактивном режиме, возможна смена режима отображения: амплитудная характеристика или фазовая.

10.04.2016 Реализован отдельный класс-хранилище объектов неопределенной длины nomain/Rdata. Его создание происходит в момент инициализации статических методов класса Memory. Так как класс Memory глобальный, то к объекту dLink, класса Rdata можно обращаться через его статические методы из любого класса приложения. Для обращения из любых потоков этот класс в своих методах использует потокобезопасные функции блокировки на мьютексах. (В дальнейшем планируется перевести методы с полной блокировки на систему read/write блокировки, что должно уменьшить задержки при параллельном чтении).

1.04.2016 Реализован класс графика gr/PlotPolarization. Битовая карта, на которой основан график, позволяет отображать как отдельные пачки, так и сумму пачек. Для этого на график вынесено поле “Период накопления”. Для удобства последующего анализа данных вынесен слайдер и связанное с ним поле “Глубина”. Изменение в этом поле (или слайдере) приводит к тому, что меняется номер пачки, с которой производится вывод на экран и накопление.

15.03.2016 Вынесен в отдельный класс массив данных и настроек Memory. В классе реализованы статистические методы, для того, чтобы не было необходимости распространять по всем классам ссылку на это хранилище. Предварительное заполнение данными производится в конструкторе класса MControl. В классе не реализовано сохранение настроек. Для этого объект класса MControl подписан на событие sync, вызывающий слот saveConfig. Слот saveConfig - метод класса MControl, в котором реализована выгрузка всех настроек из Memory в ini файл. Таким образом введение дополнительного элемента настроек (дополнительной переменной) и передача его в Memory автоматически добавляет его в файл настроек и не требует отдельного описания, а значит исключает возможность ошибки и потери настройки. Все переходы и переносы данных переработаны для работы с Memory.

3.02.2016 Реализована корреляционная обработка сигнала, пришедшего с метода loadSrc, класса Worker. Была попытка вынести обработку отдельных пачек в отдельный поток сначала с помощью пула потоков, потом с помощью 3-4 отдельных потоков, но большого прироста скорости не наблюдалось, поэтому принято решение оставить всю обработку именно в этом классе (Worker). В дальнейшем возможно придется вернуться к варианту с 3-4 отдельными потоками, в случае, если worker не будет успевать обрабатывать реальные данные. Предварительно скорость обработки позволяет не тратить на это дополнительные ядра процессора. Так же была проведена небольшая оптимизация корреляционной обработки в функции MyCorrelation. Она выполняется 8 раз по каждому блоку пачек, поэтому даже незначительное ускорение дает прирост скорости обработки, что очень важно для online режима.

17.10.2015 Закончена работа над 3D графиком. Данный график реализован с помощью технологии openGL. Отдельно написан виджет, через расширение класса QOpenGLWidget, а отдельно класс - хранилище изображения ScObject. При этом все основные операции производятся с ScObject, а когда изображение подготовлено к выводу, тогда вызывается метод updateSc, класса-виджета PlotGl. В объект класса ScObject подается список координат точек, а в классе происходит обработка цвета каждой точки и создание по результату октаэдра с нужным цветом. В случае изменении размера выводимой области происходит перерисовка всех точек и перерисовка прямоугольной сетки, ограничивающей график визуально. В PlotGl реализован 3D поворот графика и его масштабирование. Эти действия не производят перерисовки октаэдров, а передаются напрямую в видеокарту средствами openGL. Этим мы добились большой скорости и плавности вывода графика и уменьшения нагрузки на процессор.

8.10.2015 Реализована базовая структура программы. Размечены основные блоки и выстроена цепочка управления. Центральным объектом выступил класс MControl. Объект класса создается в GWindow, основном окне программы. Все остальные объекты создаются либо в GWindow, либо в MControl. В GWindow выделен MDI-блок под графики. Остальные окна будут либо Dock-виджетами этого окна, либо отдельными окнами, но только в виде исключения. Само окно GWindow пустое. Кроме MDI-блока, меню и статус-бара в нем не будет элементов управления.

Так же планируется сделать систему плагинов, которые будут менять логику обработки данных.