1. Фильтр Калмана для RSSI

Применение фильтра Калмана для «сырых» RSSI предполагает решение двух задач:

- сглаживание флуктуирующих значений RSSI;
- восстановление «пропущеных» пакетов показаниями модели фильтра Калмана.

В результате применения фильтра ожидается более эффективное сглаживание (шумоподавление) принятых значений RSSI и восстановление пропущенных пакетов на основе модели фильтра Калмана.

В основу фильтра положена модель равномерного изменения RSSI от времени, которая записывается в следующем виде:

$$\begin{cases} RSSI_i = RSSI_{i-1} + \Delta t \cdot \Delta RSSI_{i-1} \\ \Delta RSSI_i = \Delta RSSI_{i-1} \end{cases}$$

где $\Delta RSSI_i$ – скорость изменения RSSI, $\Delta RSSI_i = RSSI_i - RSSI_{i-1}$.

2. Основные уравнения

Под реализацией фильтра Калмана будем понимать формирование всех матриц, необходимых для его использования (рисунок ниже).

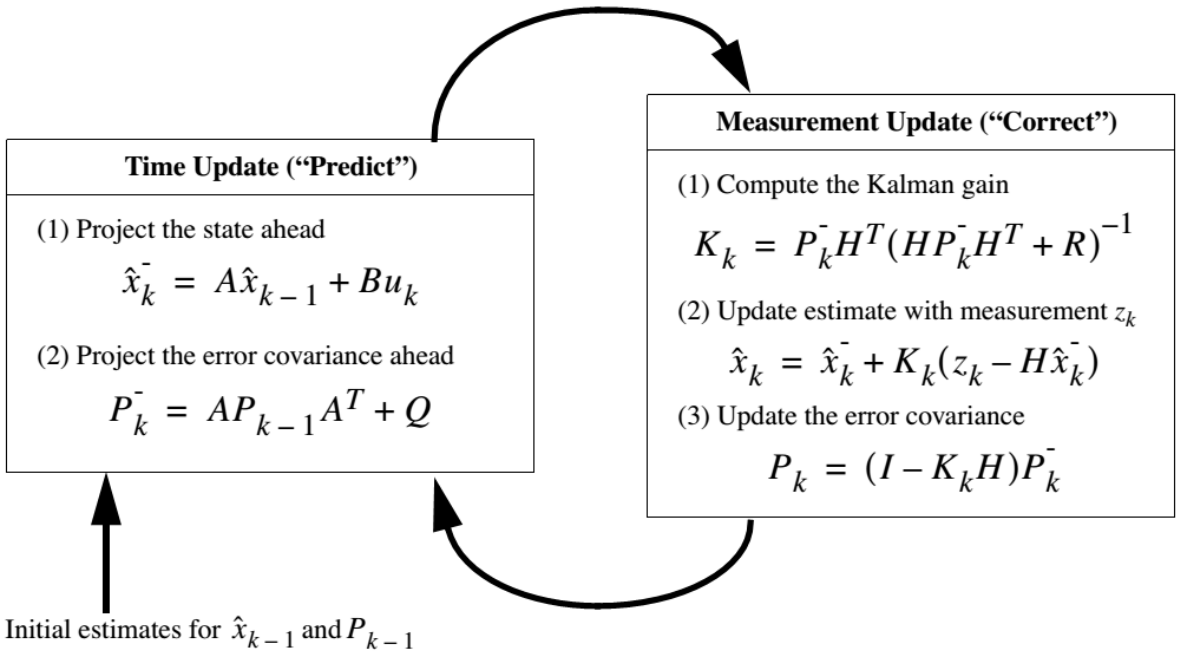

Рисунок 1 – Общая схема работы фильтра Калмана

Фильтрация осуществляется в несколько этапов:

1) ***Предсказание*** ("Predict" на рис. выше). Этап заключается: **а)** в предсказании значений $RSSI_k$ и $\Delta RSSI_k$ для текущего момента времени k (вектор X_k^-) по модели фильтра с использованием значения RSSI за предыдущий момент времени k-1 (вектор X_{k-1}); **б)** в предсказании значения матрицы ковариации ошибок модели (P_k^-) с использованием значения матрицы за предыдущий момент времени k-1 (P_{k-1}) и матрицы шума модели фильтра (Q).

2) ***Коррекция*** ("Correct" на рис. выше). Этап заключается:

а) в вычислении коэффициента Калмана K_k (по сути, он показывает, в какой пропорции доверять предсказанным значениям $RSSI_k$ и $\Delta RSSI_k$ по модели (X_k^-) и измеренному значению RSSI (Z_k));

б) в вычислении значений $RSSI_k$ и $\Delta RSSI_k$ для текущего момента времени k (X_k) с учетом предсказанных значений по модели фильтра (X_k^-), измеренного значения RSSI (Z_k) и коэффициента Калмана K_k (другими словами, в коррекции предсказанных значений (X_k^-) по измерению RSSI (Z_k), где степень коррекции определяется коэффициентом Калмана (K_k));

в) в вычислении значения матрицы ковариации ошибок модели для текущего момента времени (P_k) по ее предсказанному значению (P_k^-) и коэффициенту Калмана K_k (другими словами, в коррекции предсказанных значений (P_k^-), где степень коррекции определяется коэффициентом Калмана (K_k)).

3. Основные уравнения

<u>Уравнение 1:</u>

$$X_k^- = A_k \cdot X_{k-1} + B \cdot U_k,$$

где

$$X_{k-1} = \begin{bmatrix} RSSI_{k-1} \\ \Delta RSSI_{k-1} \end{bmatrix};$$

$$\mathrm{A}_k = \begin{bmatrix} 1 & \Delta t_k \\ 0 & 1 \end{bmatrix},$$

$$\mathrm{B} = \mathrm{U}_K = 0;$$

Δt_k – интервал времени между приходом двух пакетов (k-го и (k-1)-го) от маячка;

Уравнение 2:

$$P_k^- = A_k \cdot P_{k-1} \cdot A_k^T + Q,$$

где $P_k = \begin{bmatrix} p_{11}(k) & p_{12}(k) \\ p_{21}(k) & p_{22}(k) \end{bmatrix}$ – ковариационная матрица, пересчитывается на каждой итерации; для инициализации используются следующие ее значения $P_0 = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix}$, которые взяты из рекомендаций в научной статье.

$Q = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$ – матрица шума модели процесса; показывает, насколько точна наша модель; считаем, что модель достаточно точна; значение "0" установить нельзя, так как в этом случае фильтр выродится;

"·" – операция матричного умножения.

Уравнение 3:

$$K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R)^{-1},$$

$R = [0.1]$ – матрица шума измерения RSSI;
$H = [1 \quad 0]$ – матрица соответствия (идентичности).

Уравнение 4:

$$X_k = X_k^- + K_k \cdot (Z_k - H \cdot X_k^-),$$

где Z_k – текущее измеренное значение RSSI.

Уравнение 5:

$$P_k = (I - K_k \cdot H) \cdot P_k^-,$$

где I = $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ – единичная матрица.

4. Начало работы фильтра Калмана

По умолчанию вектор **X** и матрица **P** инициализированы нулями. Когда приходит первый пакет от определенного маячка, т.е. **Z**(k)!=0, то на блок-схеме на рисунке 5 выполняется условие:

RSSI(bN)!=0 && (**X_many**(1,bN) == 0) && (sum(**P_many**(:,bN)) == 0).

Далее выходное значение фильтра X_k без его (фильтра) запуска приравнивается к X_k=[**Z**(k); 0]^T, а значения матрицы P_k приравниваются значениям P_0 (или P_{init} по блок-схеме), Δt=0.

На следующем шаге, k+1, фильтр работает так, как это описано пятью выражениями выше (Δt=t_{k+1} - t_k).

Если промежуток времени с момента прихода последнего пакета от маячка не превышает *timeout*1 (по умолчанию равно 1,5 сек.), т.е. срабатывает условие (**Z**(k)==0 && $X_k(1)$!=0), то фильтр Калмана можем не запускать, а расчеты упростить и сделать следующим образом:

$$X_k = A_k \cdot X_{k-1},$$
$$P_k = A_k \cdot P_{k-1} \cdot A_k^T + Q$$

Если промежуток времени с момента прихода последнего пакета от маячка превысил *timeout*1 (по умолчанию равно 1,5 сек.), но меньше *timeout*2 (по умолчанию равно 5 сек.), т.е. срабатывает условие (**Z**(k)==0 && $X_k(1)$!=0), то фильтр Калмана не запускается, а параметры фильтра рассчитываются следующим образом:

$$X_k = X_{k-1},$$
$$P_k = P_{k-1}$$

т.е. просто дублируем последнее показание.

Если промежуток времени с момента прихода последнего пакета от маячка превысил *timeout*2 (по умолчанию равно 5 сек.), то фильтр Калмана для данного маячка выключается, а параметры фильтра обнуляются как при старте:

$$X_k = (0\ 0)^T,$$
$$P_k = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

5. Блок-схема реализации фильтра (с учетом всех маячков)

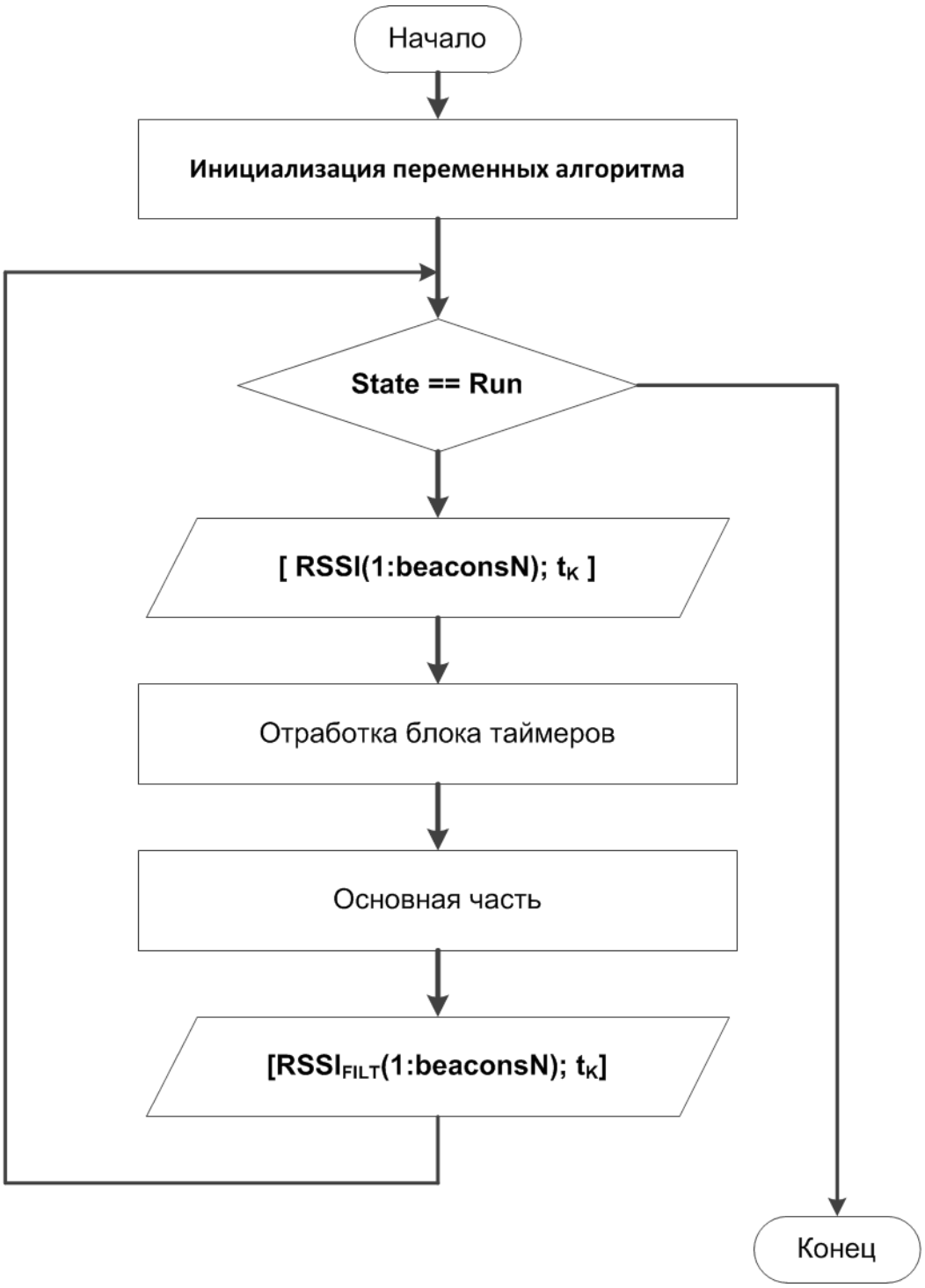

Рисунок 2 - Общий поток этапов алгоритма

Инициализация переменных алгоритма

В алгоритме используются следующие глобальные переменные:

beaconsN – Количество маячков
X_many – массив (размером 2 строки х **beaconsN** столбцов), который в столбце для каждого маячка хранит параметры (RSSI(t); ΔRSSI(t)); инициализируется нулями; используется для хранения значений матрицы **X** для каждого маячка;
P_many – массив (размером 4 строки х **beaconsN** столбцов) состояний ковариационной матрицы; инициализируется нулями; используется для хранения значений матрицы **P** для каждого маячка;
timer – массив (размером 1 строка х **beaconsN** столбцов) счетчиков; хранит информацию о том, как давно было получено последнее измерение для определенного маячка.

В алгоритме используются следующие константы и параметры:

P_init – массив (матрица 2 х 2) начальных значений ковариационной матрицы; значения: (1я строка) (100, 0) и (2я строка) (0, 100);
Q – массив (матрица 2 х 2) значений дисперсии шума модели процесса; значения: (1я строка) (0.001, 0) и (2я строка) (0, 0.001);
R – значение дисперсии шума измерений; равно 0.1;
timeout1 – время, в течении которого потерянные пакеты восстанавливаются при помощи модели; если новых пакетов от маячка за данное время не пришло, то далее используется последнее сохраненное значение на выходе фильтра; по умолчанию равно 1.5 сек.
timeout2 – время, в течении которого потерянные пакеты восстанавливаются при помощи дублирования последнего пакета на выходе фильтра при t<timeout1; если новых пакетов от маячка за данное время не пришло, то далее фильтр выключается; по умолчанию равно 5 сек.
H – матрица идентичности (матрица-строка на 2 элемента); значения равны (1, 0);
A – матрица преобразования (матрица 2 х 2); значения равны: (1я строка) (1, Δt) и (2я строка) (0, 1); значение Δt (интервал времени между предыдущим и текущим пакетами) меняется в процессе работы алгоритма;
I – единичная матрица (матрица 2 х 2) с единицами на главной диагонали, т.е. значения равны: (1я строка) (1, 0) и (2я строка) (0, 1);

В алгоритме используются следующие локальные переменные:

X – массив (размером 2 ячейки), который для текущего маячка хранит параметры (RSSI(t); ΔRSSI(t)); инициализируется нулями;
P – массив (размером 2 строки х 2 столбца) текущих значений ковариационной матрицы для рассматриваемого маячка

Рисунок 3 – Инициализация переменных

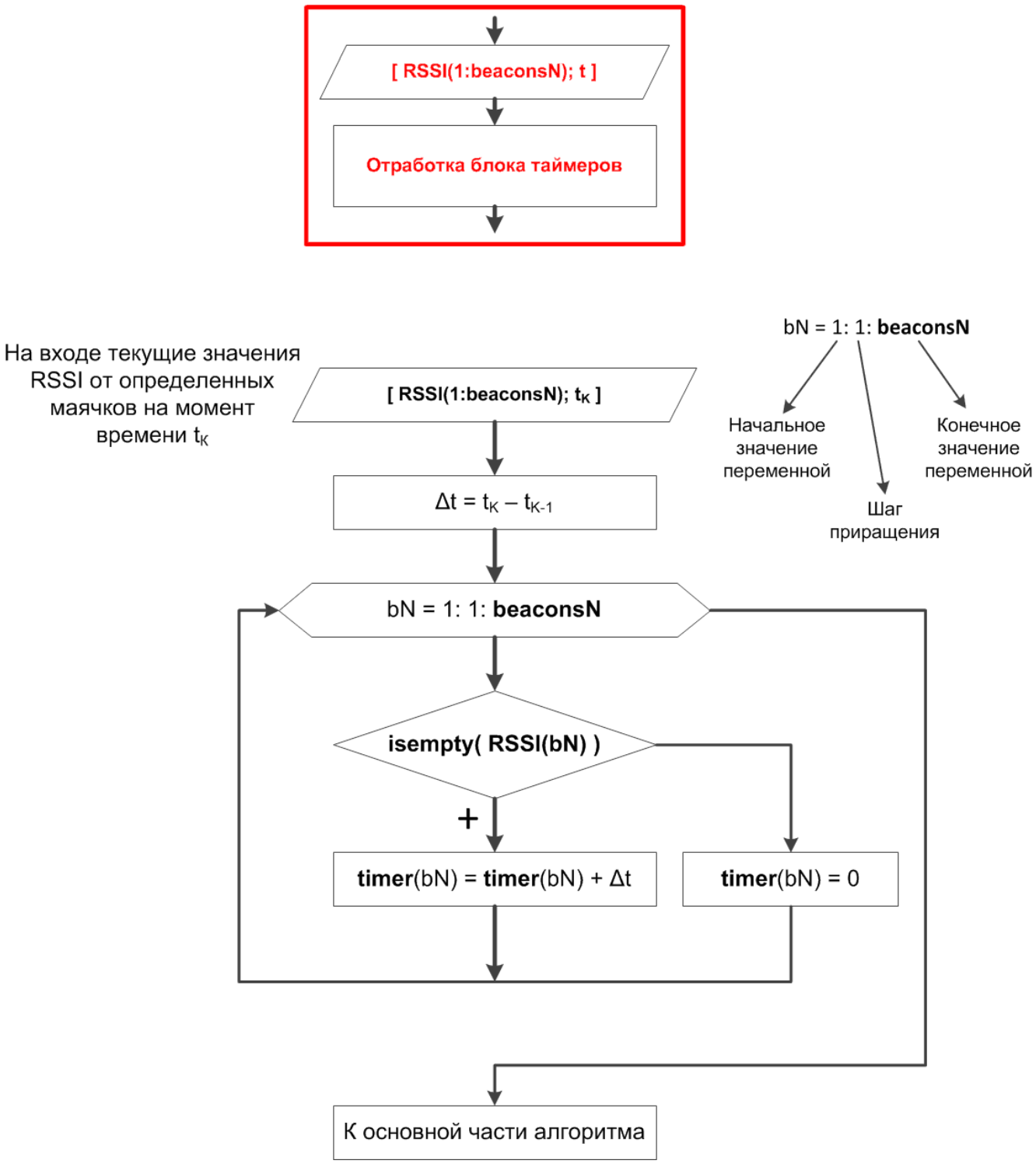

Рисунок 4 – Блок таймеров

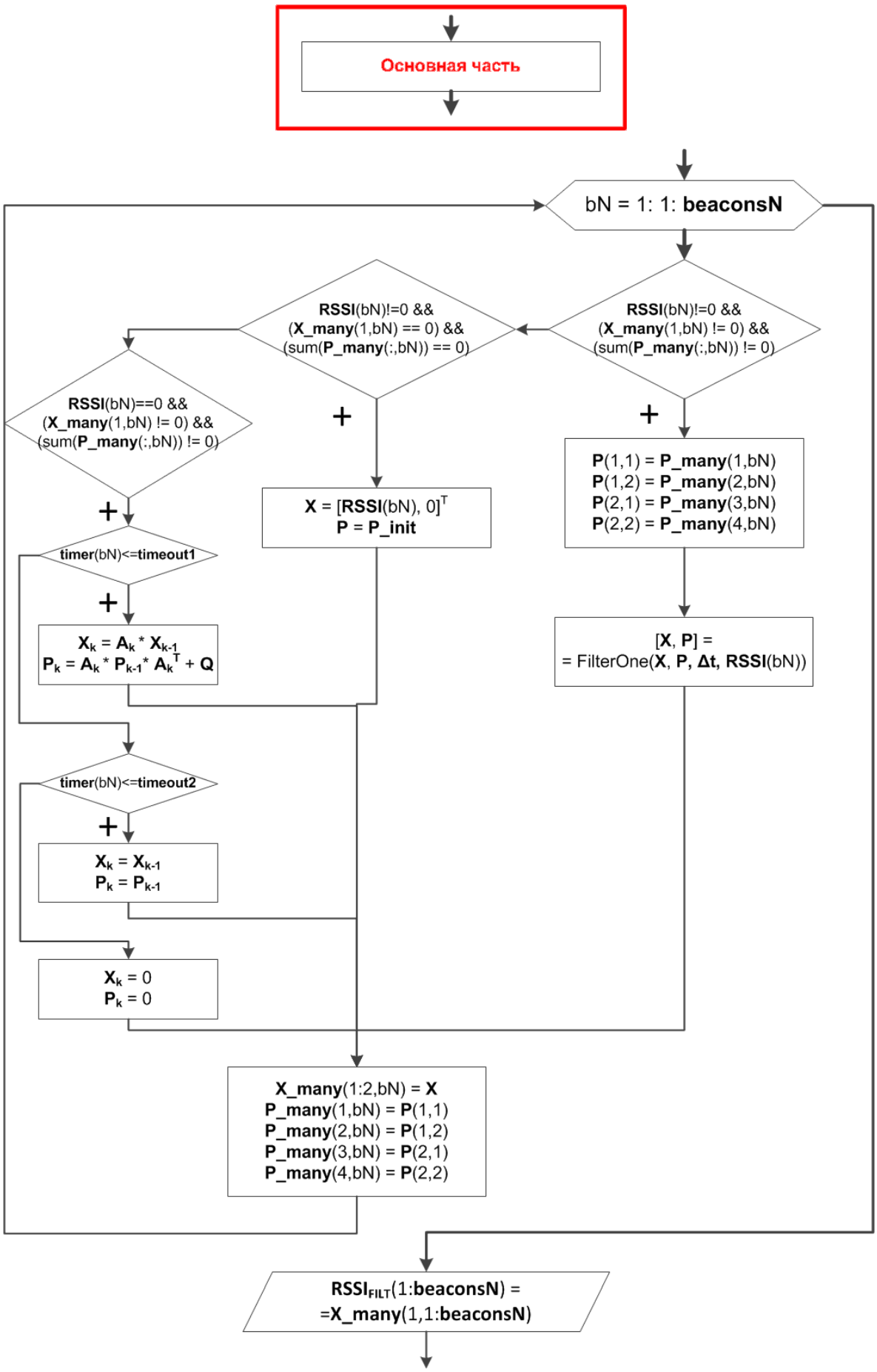

Рисунок 5 – Основная часть

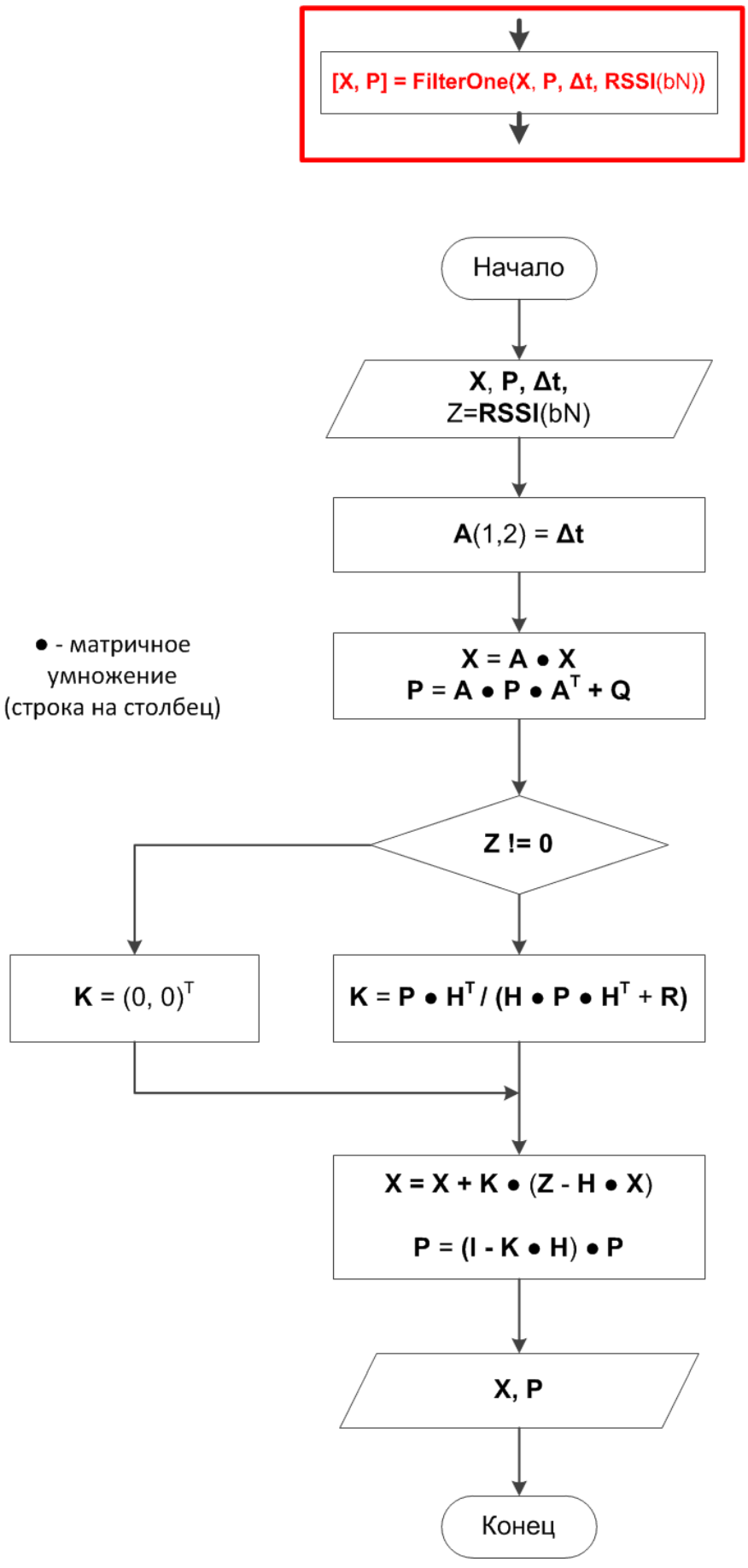

Рисунок 6 – Функция FilterOne

Поскольку фильтр Калмана довольно распространен в задачах трекинга и управления, и в Indoor, скорее всего, будет использовано две его реализации (фильтр для RSSI – текущий, и для координат – в будущем), возможно имеет смысл создать виртуальный класс, от которого затем наследовать конкретные реализации. Это будет достаточно удобно и эффективно, поскольку уравнения в фильтре в любом случае остаются неизменными.