**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Кубанский Государственный Университет**

Кафедра информационных технологий

ОТЧЕТ

о выполнении лабораторной работы №5

по дисциплине «Системы реального времени»

Выполнил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр. 44 Ерюшев И.Д. \_\_

Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. ИТ Полетайкин А.Н.

Краснодар

2024

**Организация обмена между датчиками, УВМ и исполнительными устройствами**

Цель: изучение принципов организации инфообмена между ядром СРВ и периферийными устройствами; приобретение практических навыков разработки функциональных схем, алгоритмов и управляющих программ для управления технологическими процессами с использованием измерительных преобразователей и исполнительных устройств.

**Задание**

Задан технологический процесс или функционирование объекта управления, контроль за состоянием которого осуществляется с помощью аналоговых и дискретных датчиков. Для аналоговых датчиков определены допустимый диапазон измерений, диапазон выходного сигнала, инерционность и интервал или режим опроса. Известен АЦП, преобразующий входную аналоговую информацию в двоичный код. В соответствии с измеряемыми значениями необходимо выдать аналоговое управляющее воздействие на исполнительные устройства, регулирующие протекание технологического процесса или состояние объекта управления. Для вывода аналоговой информации используется ЦАП. Разрядности АЦП и ЦАП, диапазоны аналоговых сигналов и максимальное время преобразования ЦАП заданы в таблице. Варианты технологических процессов приведены в приложении. По согласованию с преподавателем технологический процесс может быть скорректирован.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № вар. | Разрядность АЦП | Разрядность ЦАП | Диапазон напряжения, В | | Макс. время преобразования ЦАП, мс |
| Входной сигнал | Выходное воздействие |
| 4 | 16 | 10 | 0 … 24 | 0 … 43,5 | 160 |

Таблица 1.1 – варианты выбора характеристик преобразователей

Порядок выполнения:

1. Сформировать список устройств для устройства связи с объектом (УСО).

2. Построить градуировочную характеристику для каждого аналогового датчика (при наличии).

3. Рассчитать двоичные эквиваленты контрольных и управляющих непрерывных величин.

4. Определить количество, разрядность и назначение портов ввода-вывода.

5. Определить назначение отдельных разрядов портов.

6. Разработать структурную схему интерфейса связи.

7. Разработать алгоритм функционирования объекта.

8. Рассчитать параметры процедуры временной задержки (при необходимости).

9. Составить программу управления (драйвер) на языке ассемблера.

10. На языке высокого уровня разработать приложение, в графическом режиме имитирующее функционирование СРВ. Обеспечить индикацию контрольных и управляющих величин. Предусмотреть возможность оперативного изменения значений контрольных параметров объекта. Масштаб времени может быть любым.

**Заданный технологический процесс**

4. Расфасовка муки в мешки. Необходимо обеспечить непрерывную работу одного канала в системе расфасовки муки. В каждом канале имеется бункер для хранения муки и расфасовке ее в бумажные мешки по 15 и 30 кг. Наличие мешка в позиции определяется двумя ФЭ. ФЭ1 определяет факт наличия мешка, ФЭ2 – тип (высоту) мешка. Факт смены мешка фиксируется изменениями сигнала ФЭ1.

Бункер контролируется тензосистемой с диапазоном измерения от 100 до 9500 кгс и диапазоном выходного напряжения от 0 до 15 В. Бункер оснащен устройством блокирования загрузки, которое управляется дискретно. При достижении массы муки в бункере 8 т бункер блокируется через 2 с после фиксации этой массы.

Бункер оснащен дозатором с ЭМП, управляемым непрерывно подачей напряжения в диапазоне от 6 до 32 В. При напряжении ниже 6 В дозатор не работает. При напряжении 32 В дозатор работает с максимальной продуктивностью. Количество дозированной муки определяется на основе показаний тензосистемы. Величина управляющего воздействия на дозатор задана в табл. Б.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Остаток до номинальной массы, кг | Более 5 | 1…5 | 0,4…1 | 0…0,4 |
| Воздействие на дозатор | 31,5 | 18,5 | 9,5 | 6,5 |

Таблица 1.2 - Заданные управляющие воздействия на дозатор

**Ход работы**

**1 Сформированный список устройств для устройства связи с объектом**

**Список устройств для УСО:**

1. Датчики:
   1. Тензометрический датчик (аналоговый): измеряет массу муки в бункере, диапазон: 100–9500 кг. Выходной сигнал: аналоговый (напряжение 0–15 В).
   2. Датчики наличия и типа мешка (дискретные). ФЭ1: определяет наличие мешка (сигнал: 0 — мешка нет, 1 — мешок на месте). ФЭ2: определяет тип мешка (сигнал: 0 — мешок на 15 кг, 1 — мешок на 30 кг).
2. Исполнительные устройства:
   1. Дозатор с электромагнитным приводом (аналоговый): подаёт муку в мешок. Управляющий сигнал: напряжение 6–32 В.
   2. Блокировка загрузки бункера (дискретная): блокирует загрузку при превышении 8 т массы муки. Управляющий сигнал: 0 — разблокирован, 1 — блокирован.
3. Устройства преобразования и управления:
   1. АЦП (цифро-аналоговый преобразователь): Разрядность: 16 бит. Диапазон напряжения: 0–24 В (входной сигнал).
   2. ЦАП (цифро-аналоговый преобразователь): Разрядность: 10 бит. Диапазон напряжения: 0–43,5 В (выходной сигнал для дозатора).
   3. Микроконтроллер: Центральное устройство управления. Принимает данные от датчиков и генерирует управляющие сигналы.

**2. Градировочная характеристика тензометрического датчика:**

Диапазон измерений: Масса: 100–9500 кг, а напряжение: 0–15 В.

Градуировочная характеристика будет линейной:

где: U — выходное напряжение (В), M — масса муки (кг), k — коэффициент наклона (напряжение на 1 кг массы), b — смещение (0В при массе 100 кг).

1. Рассчитаем коэффициент наклона k:
2. Смещение b:

Итак, характеристика принимает вид:

Выделим ключевые точки:

1. При M=100 кг:
2. При M=9500 кг

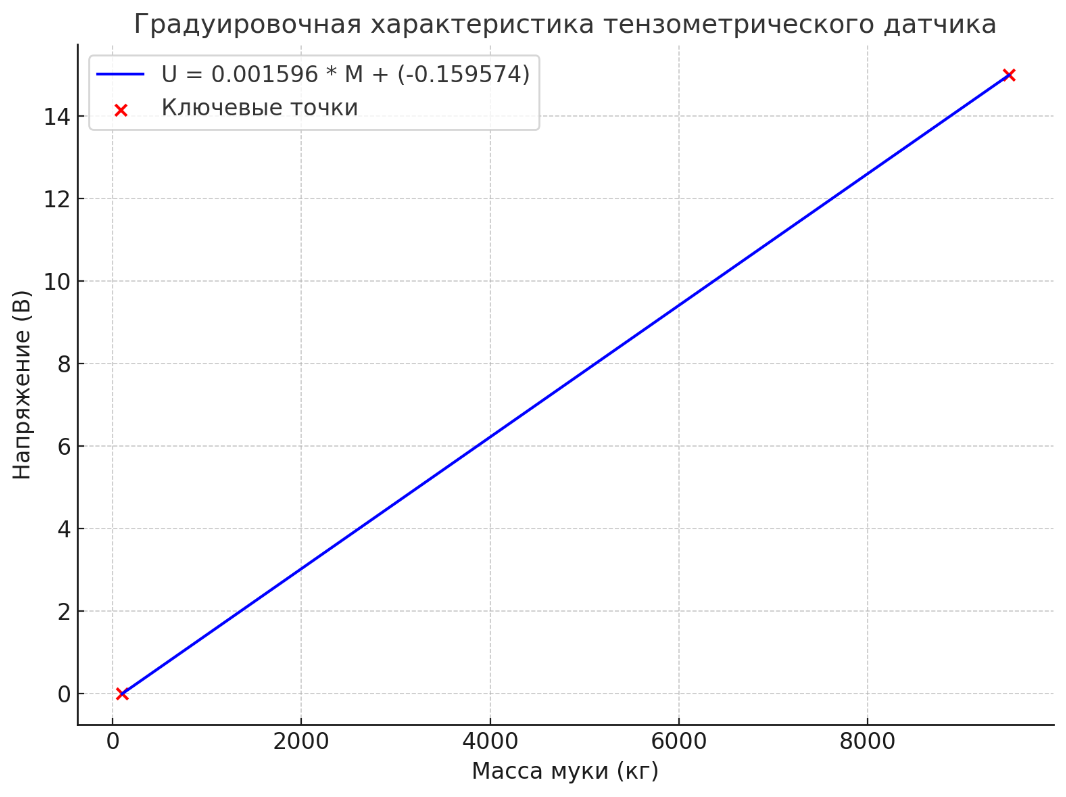


Рисунок 1 – График градуировочной характеристики тензометрического датчика

**3 Двоичные эквиваленты контрольных и управляющих непрерывных величин**

Для вычисления двоичных эквивалентов контрольных и управляющих величин используется преобразование аналоговых значений (напряжение) в цифровой код с учетом разрядности АЦП и ЦАП. Формула для вычисления двоичного кода следующая:

где: u — текущее напряжение, u1 и u2 — минимальные и максимальные значения напряжения для АЦП или ЦАП, n — разрядность АЦП или ЦАП.

Для АЦП с разрядностью 16 бит и диапазоном напряжений 0...24 В максимальный код будет равен , что соответствует максимальному напряжению 24 В. Для ЦАП с разрядностью 10 бит и диапазоном напряжений 0...43,5 В максимальный код будет равен , что соответствует максимальному напряжению 43,5 В.

| U, В (АЦП) | D (АЦП), код | U, В (ЦАП) | D (ЦАП), код |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 5 | 13653 | 5 | 115 |
| 10 | 27307 | 10 | 230 |
| 15 | 40960 | 15 | 345 |
| 20 | 54613 | 20 | 460 |
| 24 | 65535 | 43,5 | 1023 |

В таблице ниже приведены результаты преобразования напряжений в двоичные коды для АЦП и ЦАП.

Таблица 2 – кодировка амперной величины для ЦАП и АЦП

**4-5 Количество, разрядность и назначение портов ввода-вывода (а также отдельных разрядов)**

На основании списка датчиков и устройств для системы управления расфасовкой муки распределим разряды портов следующим образом:

32-битный порт ввода (необходимые разряды):

1. Разряд 0: Датчик уровня муки в дозаторе (ФЭ1);
2. Разряд 1: Датчик наличия мешка (Фэ2);
3. Разряд 2: Зарезервированы
4. Разряд 3-18: АЦП
5. Разряд 19: Сигнал готовности от АЦП
6. Разряды 20-32 Зарезервированы

2. 16-битный порт вывода (необходимые разряды):

1. Разряд 0-9: ЦАП;
2. Разряд 10: Сигнал запуска ЦАП;
3. Разряд 11: Сигнал запуска АЦП;
4. Разряд 12: Устройство блока;
5. Разряды 13-16: Зарезервированы (не используются).

Итог: всего портов: 1 порт ввода (32-разрядный) и 1 порт вывода (16-разрядный).

**6 Структурная схема интерфейса связи**

Текстовое описание взаимодействия всех компонентов системы управления:

1. Данные с датчиков (дискретные и аналоговые) поступают в порт ввода, где они распределяются по соответствующим разрядам. Данные о текущем весе поступают от тензометрического датчика как аналоговый сигнал, преобразованный в цифровой через АЦП.
2. Микроконтроллер считывает данные с порта ввода, обрабатывает их и формирует управляющие команды. Производится вычисление текущего веса, уровня заполнения и состояния заслонки.
3. Микроконтроллер отправляет сигналы на порт вывода для управления дозатором (открытие/закрытие) и блокировкой конвейера (включение/выключение).
4. Микроконтроллер передает двоичные данные о требуемом весе в ЦАП, который преобразует их в аналоговый сигнал для точного управления дозатором.
5. Все дискретные сигналы от датчиков и управляющих устройств подключены к портам ввода и вывода. Аналоговый сигнал управляется через ЦАП для обеспечения точности дозировки.

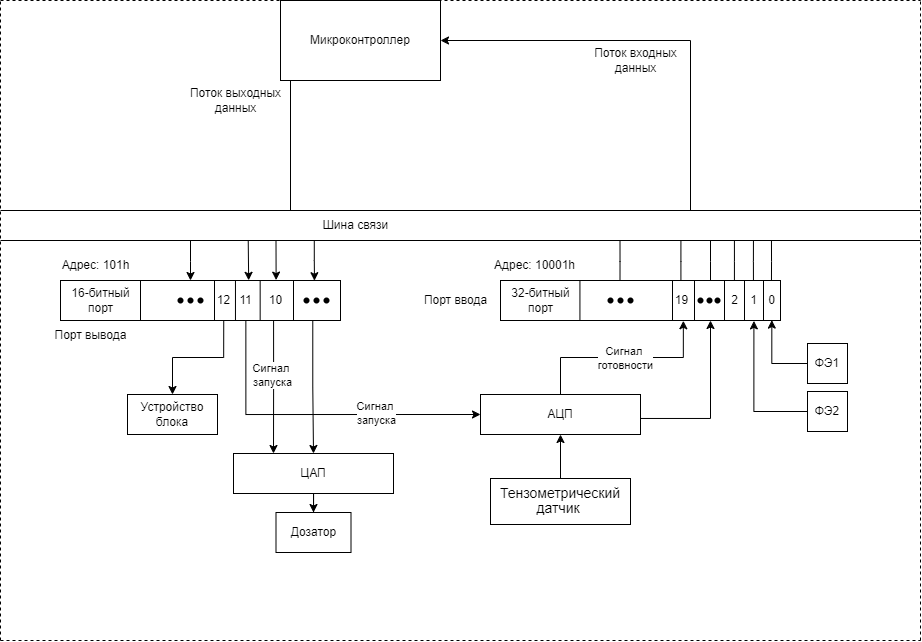


Рисунок 2 – интерфейс связи

**7 Алгоритм функционирования объекта**

1. Инициализация. Указываются адреса портов ввода-вывода. Рассчитывается время преобразования ЦАП на основе заданных параметров. Устанавливается статус системы как «ожидание».
2. Ожидание разрешающего сигнала от блока датчиков. Проверяются статусы аналоговых и дискретных датчиков: состояние блока загрузки, уровни напряжения входных сигналов от датчиков массы муки, наличие перегрузки. При выполнении всех условий система переходит к следующему этапу.
3. Управление блокировкой загрузки. На исполнительное устройство подаётся управляющий сигнал, соответствующий определённому уровню напряжения. Проверяется состояние системы через опрос всех датчиков. Если условия штатной работы сохраняются, процесс продолжается.
4. Расфасовка. Исполнительный механизм активируется, задаётся сигнал на открытие дозатора. Проводится постоянный мониторинг уровня напряжений от датчиков. При достижении заданного уровня массы муки формируется управляющее воздействие для закрытия дозатора.
5. Завершение процесса. Устройства отключаются. Блокировка загрузки возвращается в исходное положение. В случае срабатывания аварийных датчиков активируется аварийная сигнализация, система возвращается в исходное состояние.
6. Аварийная обработка. При любых сбоях (выход из диапазона напряжений, перегрузка, отсутствие сигналов) выполнение всех операций прекращается. Включается аварийная индикация, система переводится в режим ожидания до устранения неисправностей.

**9. Программа управления (драйвер) на языке ассемблера**

Драйвер системы управления реализует основные функции взаимодействия ядра системы с периферийными устройствами и обеспечивает выполнение технологического процесса. В драйвере предусмотрена работа с датчиками, исполнительными механизмами и управление состояниями системы. Код написан на языке ассемблера для архитектуры x86.

**Код программы**

.model flat

.data

port\_IN DW 100001h ; Порт для входных данных

port\_OUT DW 101h ; Порт для выходных данных

system\_status DW 0 ; Статус системы

res\_calc DW ?

; Параметры устройств

weight\_limit DW 8000 ; Лимит массы для блокировки загрузки

weight\_sensor\_signal DW 0 ; Сигнал от тензометрического датчика

bag\_presence DW 0 ; Наличие мешка (0 — мешка нет, 1 — мешок есть)

bag\_type DW 0 ; Тип мешка (0 — 15 кг, 1 — 30 кг)

dac\_signal DW 0 ; Сигнал на управление дозатором

error\_flag DW 0 ; Флаг ошибок (0 — нет, 1 — ошибка)

; Интерфейсы управления

adc\_signal DW 0 ; Цифровой сигнал с АЦП

; Проверка наличия мешка

\_check\_bag\_presence:

mov bx, bag\_presence

ret

; Проверка типа мешка

\_check\_bag\_type:

mov bx, bag\_type

ret

; Проверка лимита массы

\_check\_weight\_limit:

mov bx, weight\_sensor\_signal

cmp bx, weight\_limit

jg BLOCK\_LOADING

ret

; Установка сигнала для дозатора

\_set\_dosator\_signal:

mov bx, dac\_signal

mov dx, port\_OUT

mov ax, bx

;out dx, ax

ret

; Блокировка загрузки

\_block\_loading:

mov error\_flag, 1 ; Включаем ошибку

mov bx, 1

mov cur\_iter, bx

jmp WAITING

; Логика дозировки

\_process\_dosing:

; В BX — целевой вес

push bx

call \_set\_dosator\_voltage

; Проверка достижения веса

WAIT\_DOSING:

mov cx, weight\_sensor\_signal

sub cx, bx

jle FINISH\_DOSING ; Если достигли веса, завершаем

call \_set\_dosator\_voltage

call \_delay

jmp WAIT\_DOSING

FINISH\_DOSING:

mov dac\_signal, 0 ; Останавливаем дозатор

pop bx

ret

; Установка напряжения дозатора в зависимости от массы

\_set\_dosator\_voltage:

; Остаток до номинальной массы в кг — в bx

; Напряжение от 6.5 до 31.5 В

mov ax, bx

cmp ax, 5

jg MORE\_THAN\_5KG

cmp ax, 1

jg BETWEEN\_1\_5KG

cmp ax, 0.4

jg BETWEEN\_0\_4\_1KG

jmp LESS\_THAN\_0\_4KG

MORE\_THAN\_5KG:

mov dac\_signal, 3150 ; 31.5 В

ret

BETWEEN\_1\_5KG:

mov dac\_signal, 1850 ; 18.5 В

ret

BETWEEN\_0\_4\_1KG:

mov dac\_signal, 950 ; 9.5 В

ret

LESS\_THAN\_0\_4KG:

mov dac\_signal, 650 ; 6.5 В

ret

; Основной цикл

\_start:

MAIN\_CYCLE:

xor ecx, ecx ; Сброс регистра ECX

mov dx, port\_IN

;in ax, dx

mov ax, 0000000000110101b

mov bx, system\_status

cmp bx, 0

je WAITING

cmp bx, 1

je CATCH

cmp bx, 2

je DOSING

cmp bx, 3

je RELEASE

cmp bx, 4

je SOURCE

cmp bx, 5

je RELEASE\_LAMP

cmp bx, 6

je SOURCE\_LAMP

WAITING:

; Проверка наличия мешка

call \_check\_bag\_presence

jnz BLOCK\_LOADING

; Проверка типа мешка

call \_check\_bag\_type

jnz BLOCK\_LOADING

; Преобразуем сигнал с АЦП в напряжение

call \_convert\_adc\_to\_voltage

; Управление дозатором

call \_set\_dosator\_signal

jmp AGAIN

DOSING:

; Логика для стадии DOSING (дозировка муки)

call \_check\_bag\_presence

jnz BLOCK\_LOADING

call \_check\_bag\_type

jnz BLOCK\_LOADING

call \_check\_weight\_limit

jnz BLOCK\_LOADING

; Проверяем тип мешка:

mov bx, bag\_type

cmp bx, 0 ; 0 — мешок 15 кг

je DOSE\_15KG ; Если мешок 15 кг, дозируем 15 кг муки

cmp bx, 1 ; 1 — мешок 30 кг

je DOSE\_30KG ; Если мешок 30 кг, дозируем 30 кг муки

jmp BLOCK\_LOADING ; Если тип мешка не известен, блокируем загрузку

DOSE\_15KG:

; Логика дозирования 15 кг

mov bx, 15000 ; Масса для дозировки (15 кг)

call \_process\_dosing ; Вызов функции дозировки

jmp AGAIN ; Переход к следующему этапу

DOSE\_30KG:

; Логика дозирования 30 кг

mov bx, 30000 ; Масса для дозировки (30 кг)

call \_process\_dosing ; Вызов функции дозировки

jmp AGAIN ; Переход к следующему этапу

BLOCK\_LOADING:

; Включаем ошибку, если превышены условия

mov dx, port\_OUT

mov ax, 0000000000000000b

;out dx, ax

jmp SHUTDOWN

AGAIN:

jmp MAIN\_CYCLE

SHUTDOWN:

ret

end \_start

**Полное описание**

1. **Инициализация системы**: Программа настраивает порты ввода и вывода, устанавливает начальное состояние устройств, и подготавливает систему к работе. Начальные параметры и устройства готовы к выполнению задач.
2. **Основной цикл работы системы**: Программа переходит в основной цикл, где последовательно выполняются проверки и управляющие действия, чтобы поддерживать нормальное функционирование системы.
3. **Проверка состояния датчиков**: Программа проверяет данные от всех датчиков, чтобы удостовериться, что система готова к дальнейшей работе. Это включает проверку наличия и типа мешка, а также уровня массы муки в бункере.
4. **Управление дозатором**: если все проверки датчиков прошли успешно, программа отправляет управляющий сигнал на дозатор, чтобы начать подачу муки в мешок.
5. **Проверка условий и переход между стадиями**: Программа контролирует выполнение этапов работы, включая сбор муки, дозировку в мешок и подготовку к завершению процесса, принимая решение о переходе к следующему этапу в зависимости от состояния системы.
6. **Обработка ошибок и блокировка системы**: если обнаружена ошибка, программа активирует сигнал ошибки и может заблокировать систему, чтобы предотвратить дальнейшие проблемы. Система ожидает устранения ошибки перед продолжением работы.
7. **Завершение работы**: Программа завершает цикл, если процесс завершен успешно, или ожидает исправления ошибки, если она возникла в ходе работы.
8. **Циклическая проверка**: Программа продолжает мониторинг всех устройств, проверяя датчики и контролируя состояние системы в реальном времени, чтобы поддерживать стабильную работу системы.

**10 Высокоуровневое имитационное приложение**

Имитационное приложение представляет из себя панель управления, которая позволяет в реальном времени (имитирует работу в реальном времени) посмотреть, как СРВ ведёт себя при различных входных данных. Запрограммированный алгоритм полностью идентичен программе-драйверу, описанной в 9 пункте.

Для написания приложения использовался язык Python и графическая библиотека TKinter.

**Код программы:**

import tkinter as tk

import random

import time

class ConveyorBeltSimulation:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Симуляция расфасовки муки")

# Панель с информацией слева

self.info\_frame = tk.Frame(root)

self.info\_frame.pack(side="left", padx=20)

# Поля информации

self.type\_label = tk.Label(self.info\_frame, text="Тип мешка: null", font=("Arial", 14))

self.type\_label.pack()

self.voltage\_label = tk.Label(self.info\_frame, text="Напряжение: 0 В", font=("Arial", 14))

self.voltage\_label.pack()

self.f1\_label = tk.Label(self.info\_frame, text="Датчик ФЭ1 (наличие мешка): 0", font=("Arial", 14))

self.f1\_label.pack()

self.f2\_label = tk.Label(self.info\_frame, text="Датчик ФЭ2 (тип мешка): 0", font=("Arial", 14))

self.f2\_label.pack()

self.lock\_label = tk.Label(self.info\_frame, text="Блокировка загрузки: 0", font=("Arial", 14))

self.lock\_label.pack()

self.status\_label = tk.Label(self.info\_frame, text="Статус: ожидание", font=("Arial", 14))

self.status\_label.pack()

# Масса бункера

self.bunker\_mass\_label = tk.Label(self.info\_frame, text="Масса в бункере: 5000 кг", font=("Arial", 14))

self.bunker\_mass\_label.pack()

# Добавим новое поле для массы мешка

self.bag\_mass\_label = tk.Label(self.info\_frame, text="Масса мешка: 0 кг", font=("Arial", 14))

self.bag\_mass\_label.pack()

# Холст для анимации

self.canvas = tk.Canvas(root, width=800, height=400, bg="white")

self.canvas.pack()

# Элементы на холсте

self.belt = self.canvas.create\_rectangle(0, 200, 800, 250, fill="gray")

self.dosator = self.canvas.create\_polygon(380, 100, 420, 100, 400, 150, fill="blue")

self.bag = None

# Управление

self.start\_button = tk.Button(root, text="Старт", command=self.start\_simulation)

self.start\_button.pack(side="left")

self.stop\_button = tk.Button(root, text="Остановить", command=self.stop\_simulation)

self.stop\_button.pack(side="left")

# Переменные состояния

self.running = False

self.bag\_speed = 10

self.bag\_position = -50

self.bag\_filling = False # Состояние заполнения

self.bag\_mass = 0 # Масса мешка

self.locked = False # Блокировка загрузки

self.bunker\_mass = 5000 # Начальная масса в бункере

self.fill\_time = 0 # Время для заполнения мешка (в секундах)

self.start\_time = 0 # Время начала заполнения

# Функция обновления

self.update\_animation()

def start\_simulation(self):

if not self.running:

self.running = True

if self.bag is None or self.bag\_position > 800:

self.spawn\_bag()

def stop\_simulation(self):

self.running = False

def spawn\_bag(self):

self.bag\_position = -50

bag\_height = random.choice([30, 50]) # Высота для определения типа мешка

self.bag = self.canvas.create\_rectangle(

self.bag\_position, 200 - bag\_height, self.bag\_position + 50, 200, fill="brown"

)

self.type\_label.config(text="Тип мешка: null")

self.voltage\_label.config(text="Напряжение: 0 В")

# Обновление датчиков

self.f1\_label.config(text="Датчик ФЭ1 (наличие мешка): 1")

self.f2\_label.config(text="Датчик ФЭ2 (тип мешка): 0") # Пока 15 кг

self.status\_label.config(text="Статус: ожидание")

def update\_animation(self):

if self.running and self.bag:

if not self.bag\_filling: # Если мешок не заполняется

self.bag\_position += self.bag\_speed

self.canvas.move(self.bag, self.bag\_speed, 0)

# Проверка под дозатором

if 375 <= self.bag\_position <= 380 and self.bunker\_mass > 0:

self.bag\_filling = True

bag\_coords = self.canvas.coords(self.bag)

bag\_height = bag\_coords[3] - bag\_coords[1]

bag\_type = "30 кг" if bag\_height == 50 else "15 кг"

voltage = 32 if bag\_type == "30 кг" else 16

self.type\_label.config(text=f"Тип мешка: {bag\_type}")

self.voltage\_label.config(text=f"Напряжение: {voltage} В")

self.f2\_label.config(text=f"Датчик ФЭ2 (тип мешка): {1 if bag\_type == '30 кг' else 0}")

self.status\_label.config(text="Статус: заполнение")

# Инициализация массы мешка

self.bag\_mass = 0

self.target\_bag\_mass = 30 if bag\_type == "30 кг" else 15

self.start\_time = time.time()

# Обновляем массу мешка в интерфейсе

self.bag\_mass\_label.config(text=f"Масса мешка: {self.bag\_mass} кг")

# Заполнение мешка и уменьшение массы из бункера

if self.bag\_filling:

elapsed\_time = time.time() - self.start\_time

fill\_rate = 0.3 # Ускоряем скорость наполнения (кг в секунду)

# Плавное увеличение массы мешка и уменьшение массы в бункере

if self.bag\_mass < self.target\_bag\_mass and self.bunker\_mass > 0:

self.bag\_mass += fill\_rate

self.bunker\_mass -= fill\_rate

self.start\_time = time.time() # сброс таймера для плавности

# Обновляем массу мешка в интерфейсе

self.bag\_mass\_label.config(text=f"Масса мешка: {self.bag\_mass:.1f} кг")

self.bunker\_mass\_label.config(text=f"Масса в бункере: {self.bunker\_mass:.1f} кг")

# Изменение напряжения в зависимости от массы мешка

remaining\_mass = self.target\_bag\_mass - self.bag\_mass

if remaining\_mass > 5:

voltage = 31.5

elif remaining\_mass > 1:

voltage = 18.5

elif remaining\_mass > 0.4:

voltage = 9.5

else:

voltage = 6.5

self.voltage\_label.config(text=f"Напряжение: {voltage} В")

else:

self.bag\_filling = False

self.status\_label.config(text="Статус: заполнение завершено")

self.voltage\_label.config(text="Напряжение: 0 В")

# Удаление мешка за пределами экрана

if self.bag\_position > 800:

self.canvas.delete(self.bag)

self.bag = None

self.spawn\_bag() # Создание нового мешка

# Обнуление данных после удаления мешка

self.type\_label.config(text="Тип мешка: null")

self.voltage\_label.config(text="Напряжение: 0 В")

self.f2\_label.config(text="Датчик ФЭ2 (тип мешка): 0")

self.status\_label.config(text="Статус: ожидание")

self.root.after(50, self.update\_animation) # Обновление каждые 50 мс

# Запуск программы

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = ConveyorBeltSimulation(root)

root.mainloop()

**Вид программы:**

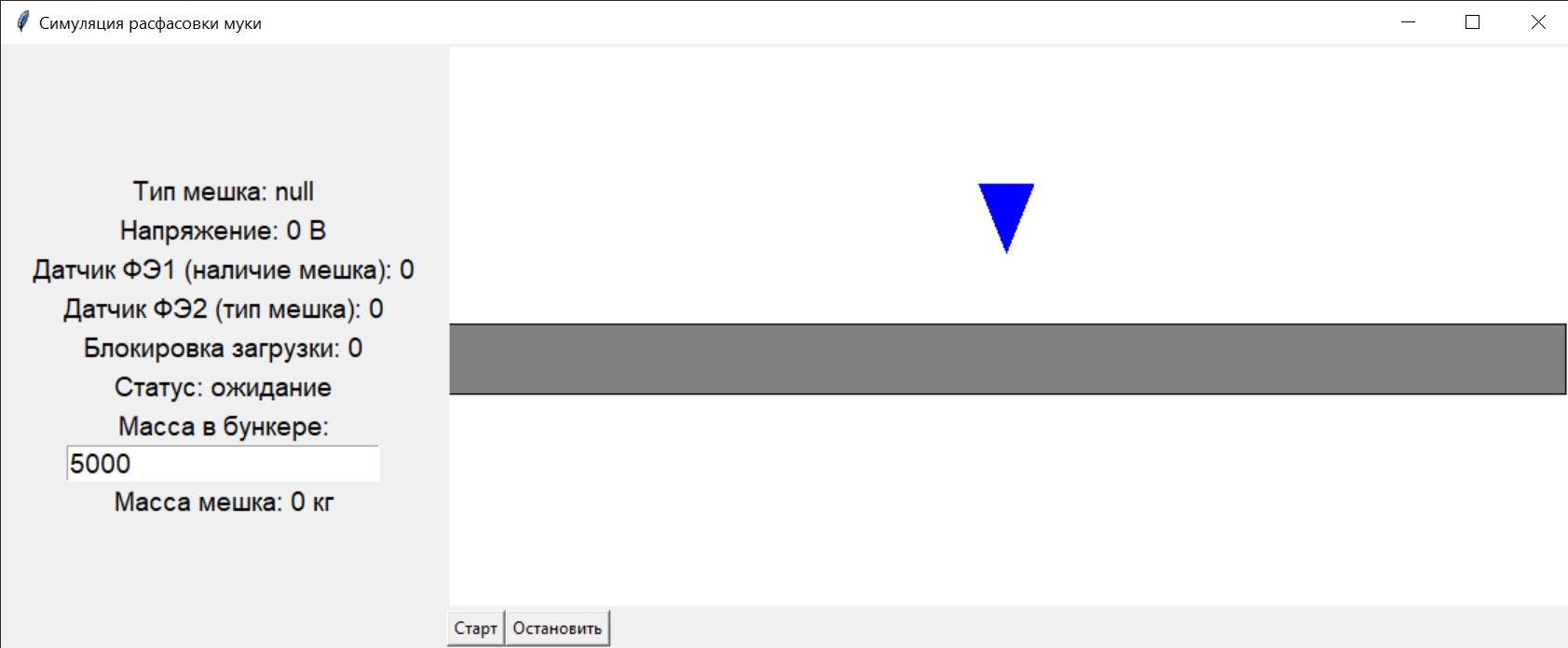


Рисунок 3.1 – Начально окно программы.

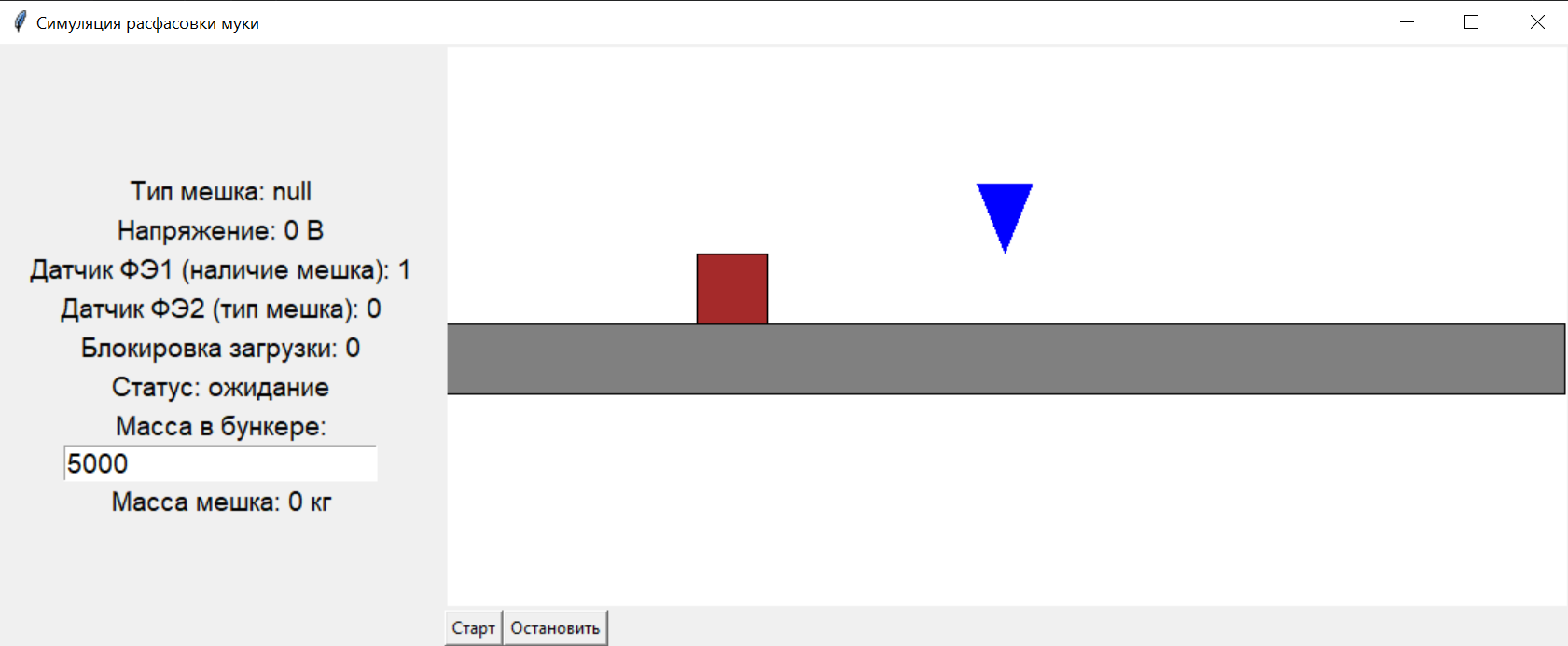


Рисунок 3.2 – Статус ожидания мешка.

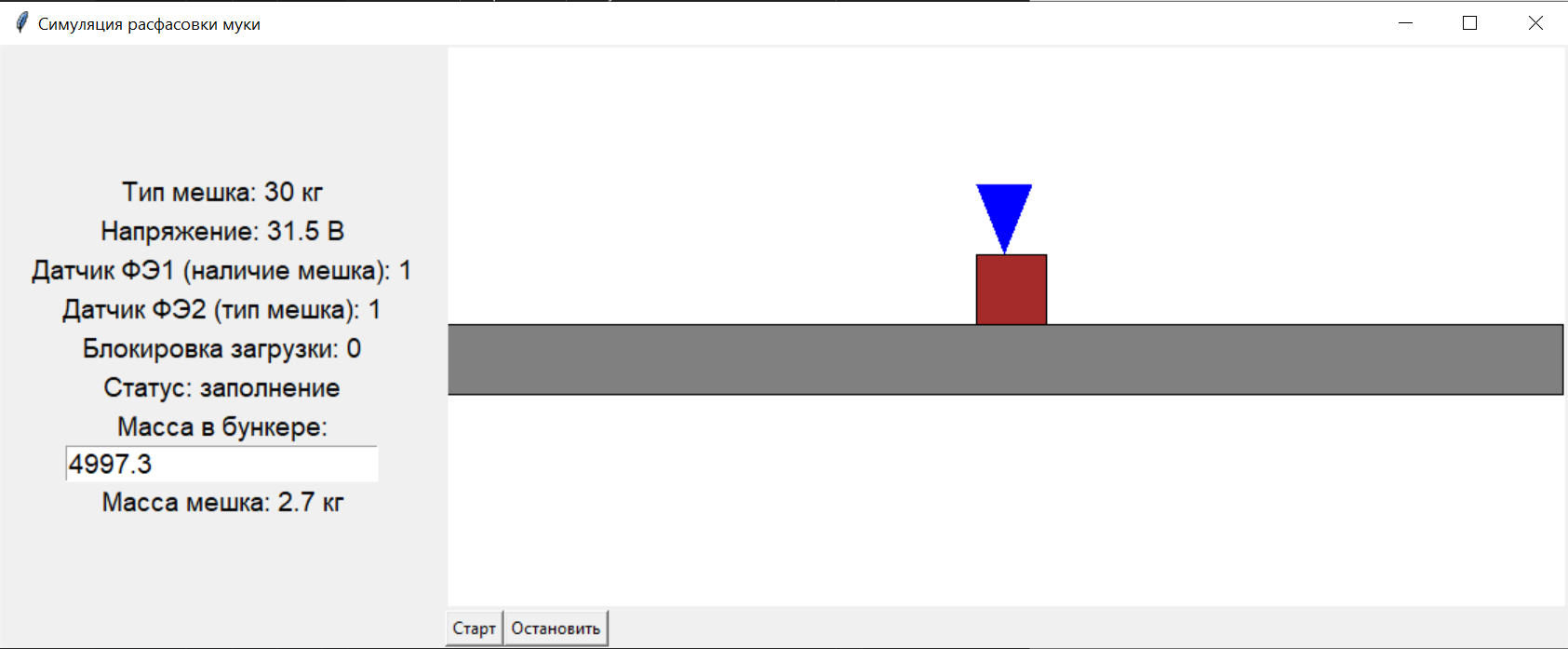


Рисунок 3.3 – Процесс заполнения мешка типа 2.

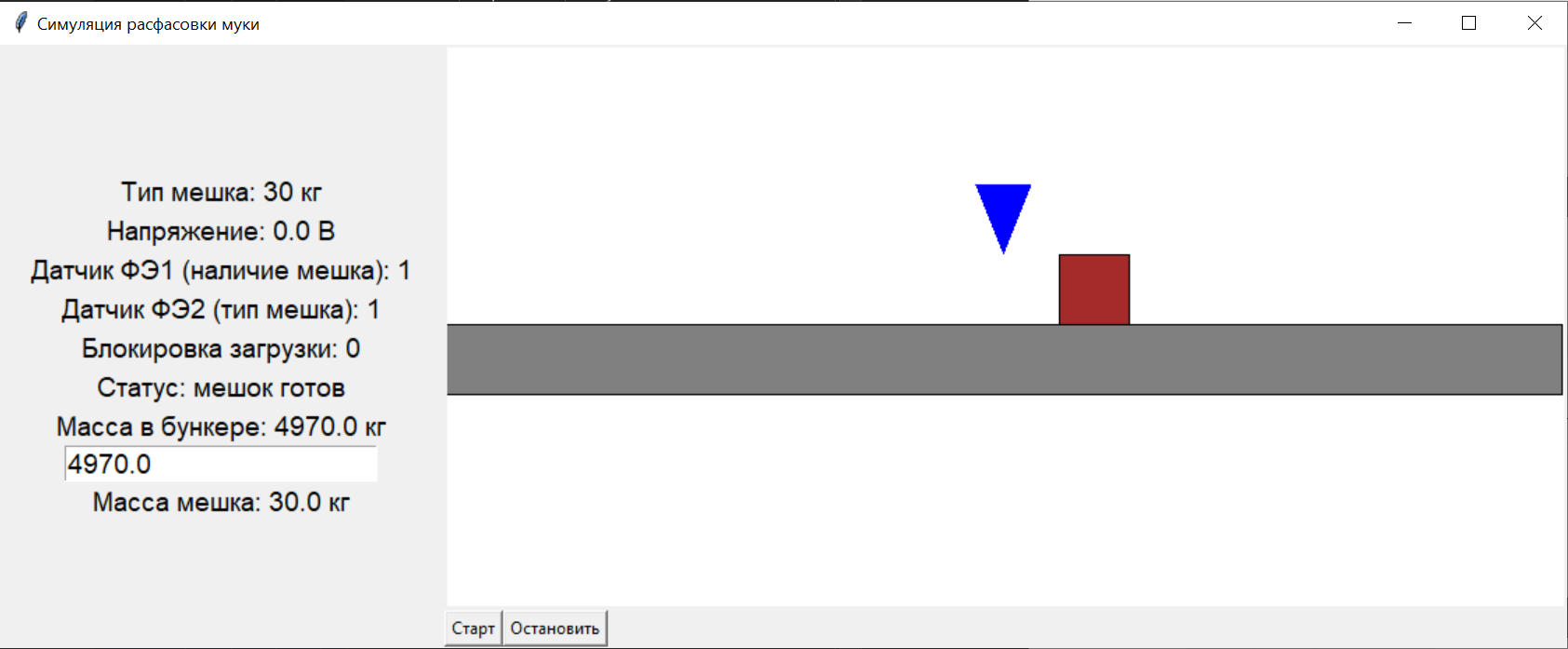


Рисунок 3.4 – Статус готовности мешка.

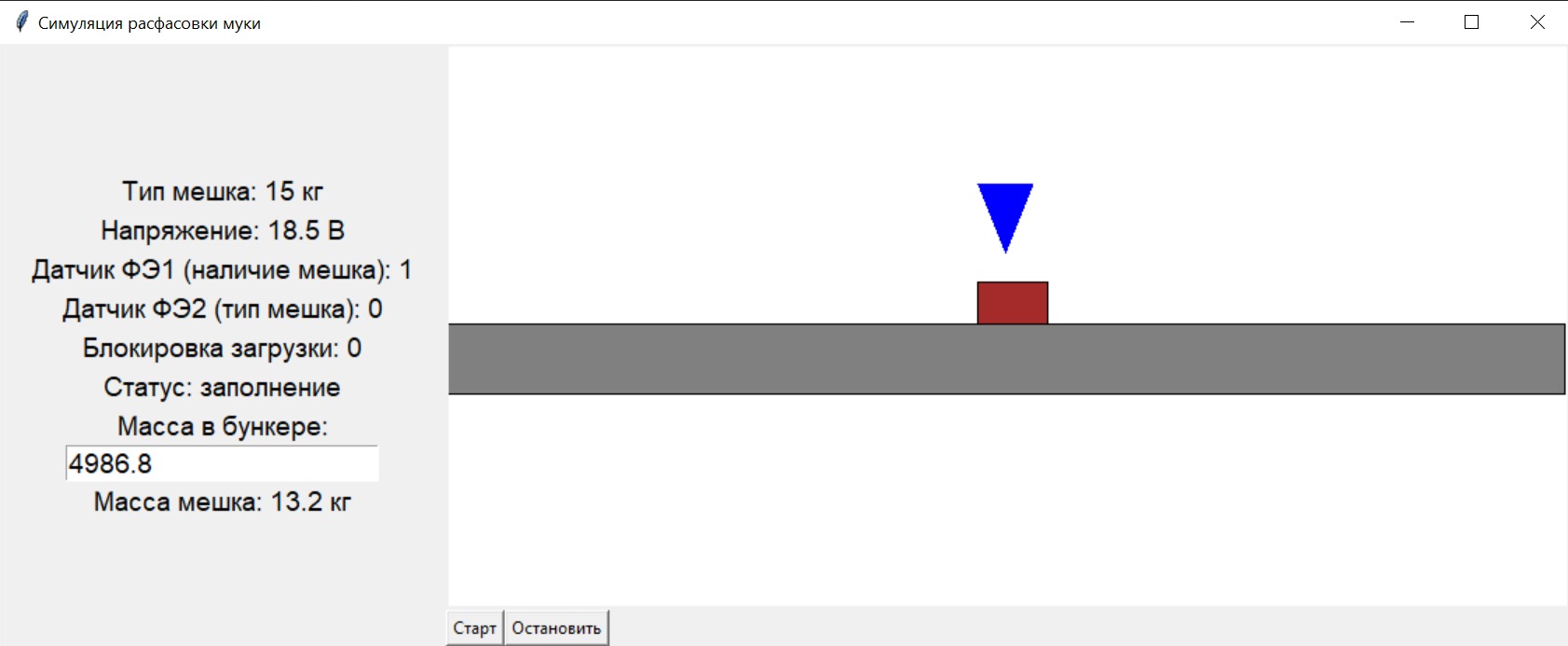


Рисунок 3.5 – Процесс заполнения мешка типа 1.

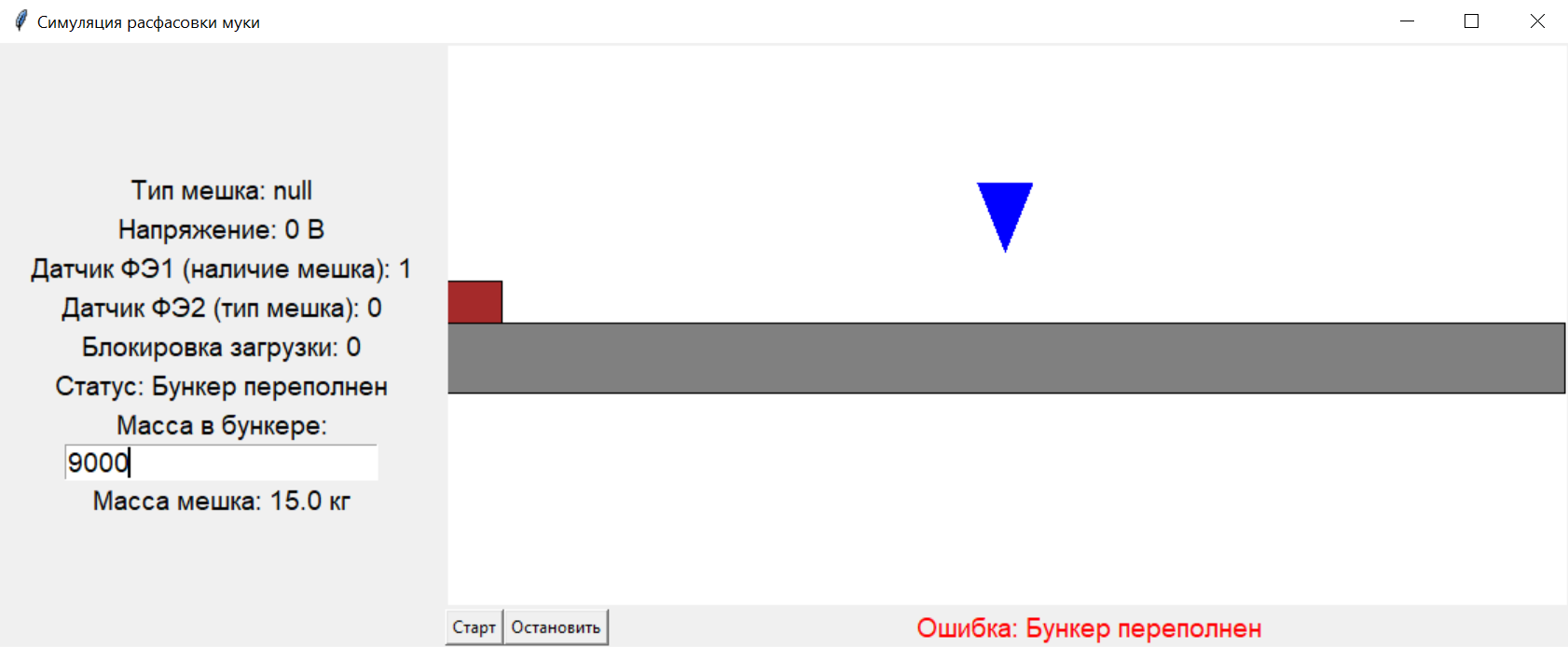


Рисунок 3.6 – Ошибка при переполнении бункера.

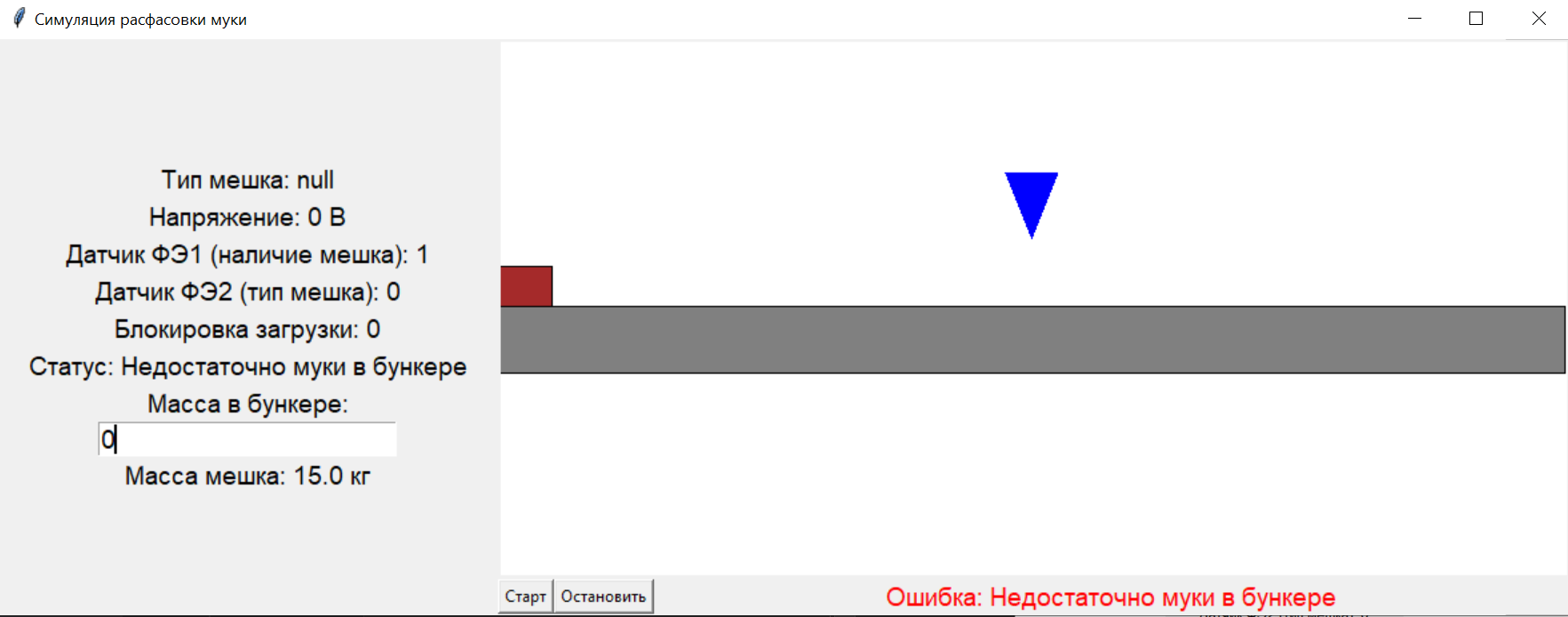


Рисунок 3.7 – Ошибка при недостатке муки в бункере.

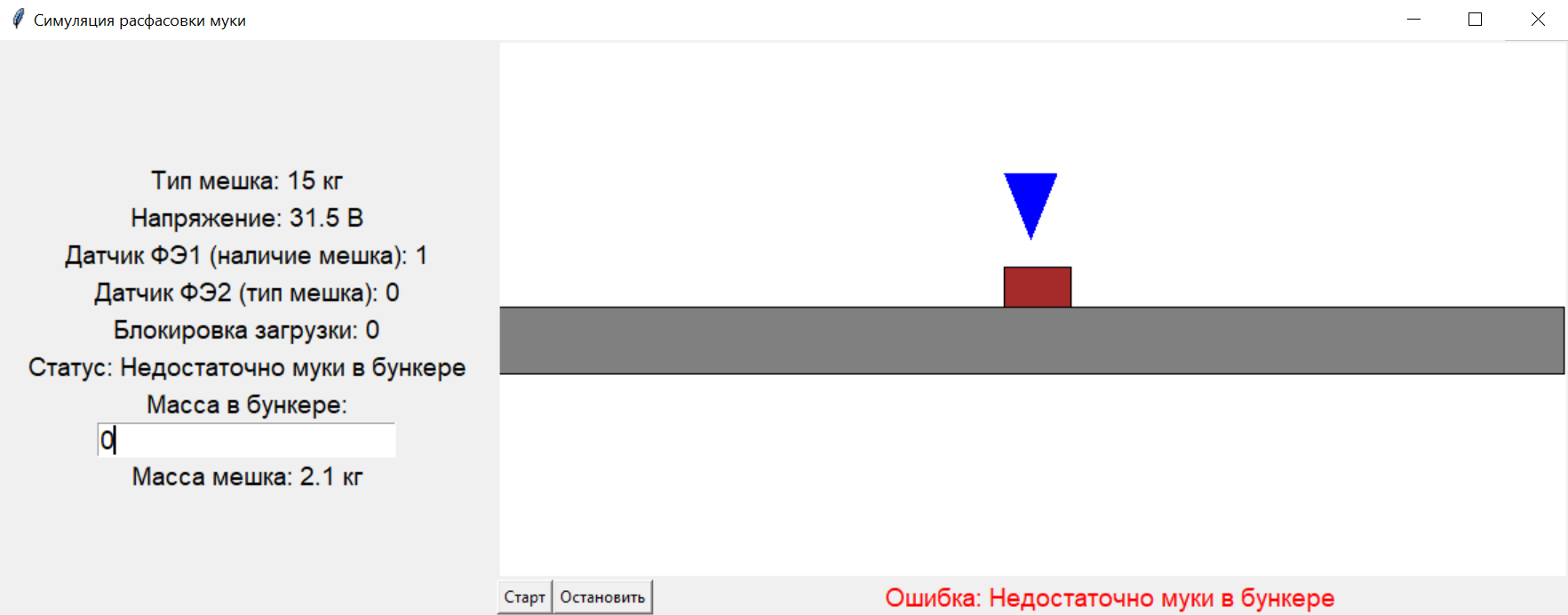


Рисунок 3.8 – Ошибка при недостатке муки в бункере,

в процессе наполнения мешка