Dr. Bob Davidov

Обработка и отображение сигналов на частоте преобразования АЦП

Цель работы: Рассмотрение особенностей ввода и отображения широкополосных сигналов.

Задача работы: Построение канала ввода, обработки и отображения сигналов на максимальной частоте преобразования АЦП контроллера Arduino.

Приборы и принадлежности: Контроллер Arduino UNO, пакет Simulink МатЛАБ (R2012).

ВВЕДЕНИЕ

Разработка программных средств для наблюдения, анализа и обработки сигналов на уровне контроллеров требует значительных временных затрат. Подключение контроллера к специализированной среде высокого уровня (Рис. 1) позволяет значительно сократить время проектирования алгоритма для контроллера с учетом ограничений его ресурсов. Хорошим примером мощной специализированной среды для работы с сигналами является МатЛАБ.

Для анализа сигналов зачастую требуется наблюдать его спектр в максимально широкой полосе частот. Для этого контроллер должен принимать сигналы на максимальной частоте преобразования АЦП.

Построение рабочего канала “Arduino UNO – МатЛАБ” для наблюдения и обработки сигналов в реальном времени на предельной частоте преобразования АЦП подробно излагается в этой работе. Особенностью этого канала является то, что такты реального времени задаются не МатЛАБ, а контроллером Arduino. Такое построение не требует компиляции Simulink модели с библиотекой реального времени (rtwin.tlc), что позволяет использовать модели с практически любыми блоками библиотеки Simulink и изменять параметры блоков во время работы модели. В отличии от традиционного режима: “одно считывание сигнала за такт реального времени”, предлагаемый вариант обеспечивает получение вектора (множества) значений сигнала на частотах сэмплирования значительно превышающих частоту реального времени. Увеличение продолжительности такта реального времени, при сохранении периода сэмплирования сигнала, позволяет применять более сложные модели с большим объемом вычислений.

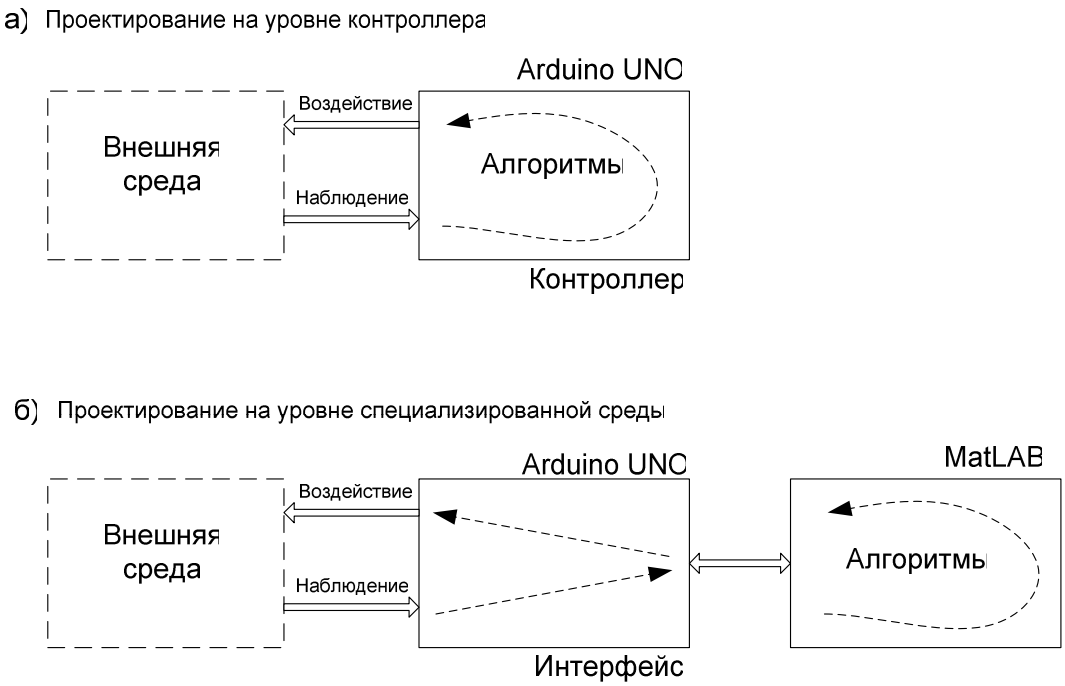

Рис. 1. Сравнение средств разработки алгоритмов. Для проектирование алгоритмов на уровне специализированной среды необходим канал передачи данных между контроллером и средой проектирования.

ОБЩИЕ СВЕДЕНИЯ

Средства накопления, анализа, обработки и отображения сигналов

В этой работе используется среда Simulink для приёма и отображения данных контроллера Arduino.

Simulink – это интерактивная среда и язык программирования для имитационного моделирования, позволяющая при помощи блок-диаграмм строить динамические модели процессов. Simulink интегрирована в среду MatLAB. Интеграция позволяет использовать уже готовые библиотеки блоков, встроенные математические алгоритмы, мощные средства обработки и графического отображения данных для решения всего спектра задач от разработки концепции модели до тестирования, проверки, генерации кода и аппаратной реализации.

Состав пакетов расширения библиотеки Simulink на примере пакета цифровой обработки сигнала “DSP System Toolbox” показан на Рис. 2.

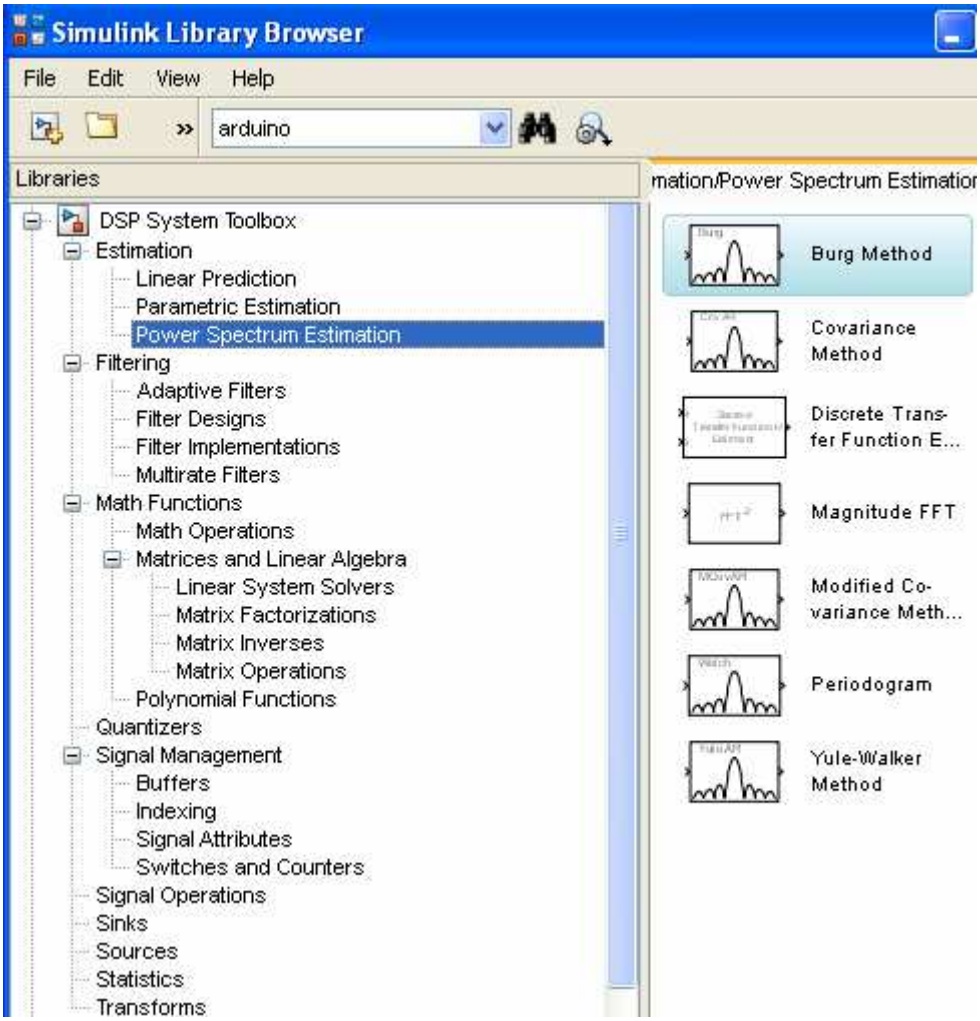

Рис. 2. Пример дополнительного пакета расширения Simulink для моделирования систем обработки сигналов: DSP System Toolbox [1]. В пакете используются новейшие алгоритмы спектрального анализа. Выделено содержимое раздела Power Spectrum Estimation - блоки для спектральной оценки сигнала.

Потоковая передача данных.

Использование двух буферов для накопления и передачи данных позволяет организовать без разрывов сбор, обработку и визуализацию данных (для избежания потери данных скорость последующего процесса должна быть не ниже скорости предыдущего процесса).

Примером применения такой организации может быть модуль E14-440 [2] для многоканального ввода, вывода и обработки аналоговой и цифровой информации, подключаемый к компьютеру через шину USB.

Входные данные сначала заносятся в первую половинку FIFO буфера АЦП. После ее заполнения данные начинают передаваться в PC, в тоже время не прекращается сбор данных во вторую половинку FIFO буфера. После накопления данных во второй половинке FIFO

буфера опять начинается передача данных в PC и параллельно продолжается сбор данных уже в первую половинку.

ПОСТРОЕНИЕ ОСЦИЛЛОГРАФА НА БАЗЕ КОНТРОЛЛЕРА ARDUINO

Максимальная скорость накопления данных АЦП

Используя вывод результата на монитор редактора Arduino на максимальной частоте (57600 бит/с) напишем программу подсчета преобразований АЦП за фиксированный период.

Программа измерения скорости преобразования АЦП:

```
const int adc_5 = A5;     // ADC port number
unsigned long time_start;   // Start of capturing, ms
unsigned long time_end;     // End of capturing, ms

void setup() {
  Serial.begin (57600); //  9600, 19200, 38400, 57600 and 115200 bit/s
}

void loop(){
  time_start = millis();
  for (int i = 0; i < 1024; i++) {
    int val = analogRead(adc_5);
  }
  time_end = millis();
  int period = time_end - time_start;
  Serial.println(period);
```

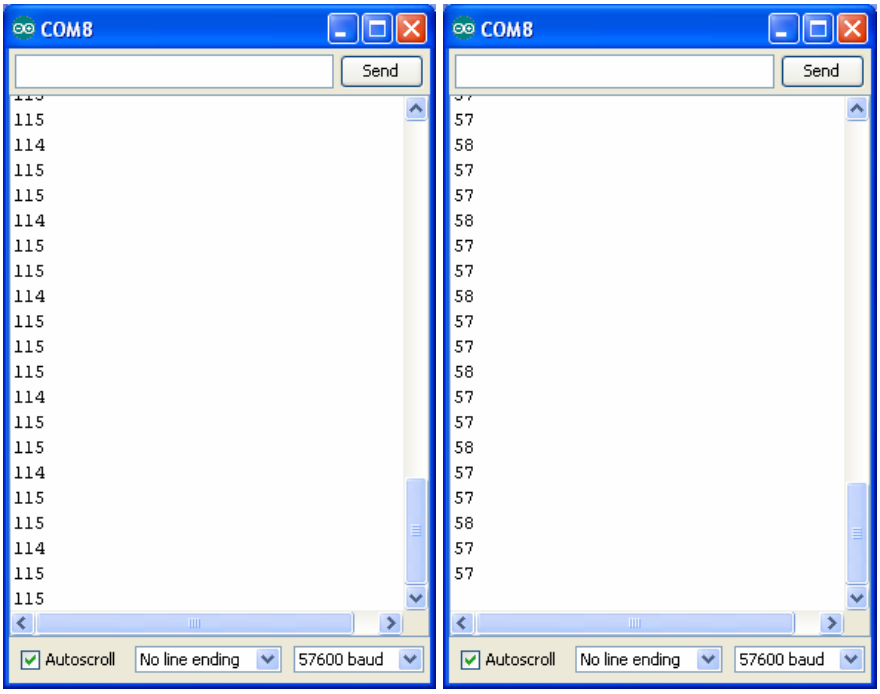

Рис. 3. Время (в мсек) 1024 и 512 преобразований АЦП. Среднее время преобразования АЦП: 0.1123 мс (как 115/1024).

Время масштабирования данных АЦП

Для перевода 10 разрядных данных АЦП в 8 разрядные используется функция

map(val, 0, 1023, 0, 255);

где val – int переменная с 10 значимыми разрядами.

Программа измерения времени преобразования АЦП с масштабированием и записи в массив:

```
const int adc_5 = A5;     // ADC port number
unsigned long time_start;   // Start of capturing, ms
unsigned long time_end;     // End of capturing, ms
byte adc_bytes[1024];   // Buffer for ADC data

void setup() {
  Serial.begin (57600); // bit/s
}

void loop(){
  time_start = millis();
  for (int i = 0; i < 1024; i++) {
    int val = analogRead(adc_5);
    adc_bytes[i] = map(val, 0, 1023, 0, 255);
  }
```

```
  time_end = millis();
  int period = time_end - time_start;
  Serial.println(period);
}
```

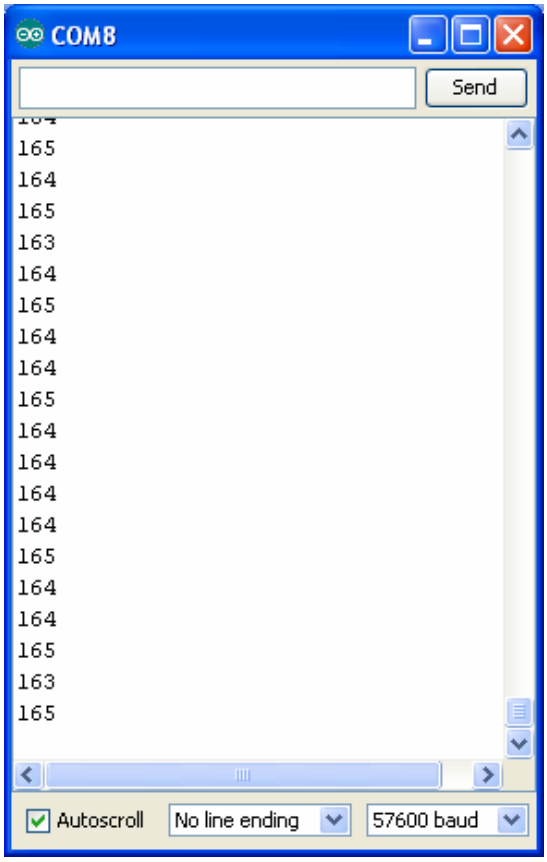

Рис. 4. Время (в мсек) 1024 преобразований АЦП, перевода 10 р. данных в 8 разрядные и запись в массив. Период АЦП преобразования с масштабированием: 0.1611 мс (как 165/1024).

Поскольку время преобразования АЦП 0.13 мс, то один перевод 10 разрядных данных в байтовый формат (масштабирование) составляет 0.0424 мс.

Скорость канала последовательной передачи данных

Для определения скорости побайтовой передачи в последовательный канал в цикле передается код символа Serial.write(1) который не отображается на мониторе.

Основной блок программы определения скорости передачи:

```
void loop(){ //Do stuff here
  unsigned long time = millis();
  Serial.write(1);
  rate = rate + 1;
```

```
  if (time > set_time) {
    set_time = set_time + 30;  // 30 ms RT clock
    Serial.println(rate);
    rate = 0;
  }
}
```

```
      176
 170
 170
169
      176
 170
169
      176
 170
169
 170
      176
169
 170
```

Рис. 5. Тестовые данные: количество переданных байт в последовательный канал за 30 мс на скорости 57600 бит/с.

Тест показал, что передача 173 байт занимает 30 мс, с другой стороны за 30 мс на скорости 57600 бит/с можно передать 1728 бит. Следовательно, на передачу одного байта расходуется время передачи 10 бит. Используя это отношение для режима передачи

- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: none

можно подсчитать время потоковой передачи массива данных на разных скоростях.

Передача, например, 256 байт на скорости 9600 бод (бит/с) занимает 267 мс, на скорости 57600 бод – 44 мс; и на скорости 115200 бод – 22 мс (как 256*10/115200).

Размер массива для накопления и передачи данных

Размер оперативной (SRAM) памяти Arduino UNO составляет 2 Кбайт. Тестирование программы циклического считывания АЦП, масштабирования 10 разрядных данных до 8 разрядных, тактирования и побайтной передачи данных показало, что максимальный размер массива для накопления и отправки данных не должен превышать 1800 байт.

Более сложные программы могут потребовать и большей дополнительной памяти SRAM. Поэтому массив для накопления и передачи данных АЦП ограничен 1024 байтами или 512 словами.

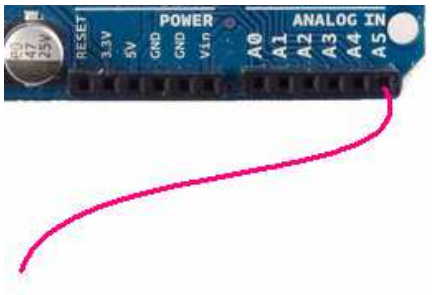

Рис. 6. Кусок провода, подсоединенный к аналоговому входу A5 контроллера Arduino для усиления наблюдаемой наводки сети 50 Гц.

Таблица 1. Времена операций программы с учетом нестабильности циклов

Длина массива, байт / слов	Время преобразования АЦП, мс	Время преобразования АЦП с масштабированием 10 в 8 бит, мс	Время передачи массива на скорости 115200 бит/с, мс	Время цикла программы
256 байт		42	23	70 (65+5)
512 байт		83	45	140 (128+12)
768		110	67	190 (177+13)
1024 байт		165	89	270 (254+16)
512 слов	58		89	160 (147+13)

Пример настройки канала отображения 256 масштабированных значений АЦП при максимальной скорости накопления и передачи данных.

Код программы контроллера Arduino:

```
const int adc_5 = A5; // ADC port number
unsigned long set_time; //  Time of next clock
byte adc_bytes[256];  // Buffer for scaled ADC data

void setup() {
  Serial.begin (115200); // bit/s
}

void loop(){
  unsigned long time = millis();   // Current time in ms

  // ADC data capturing
  for (int i = 0; i < 256; i++) {
```

```
  int val = analogRead(adc_5);
  adc_bytes[i] = map(val, 0, 1023, 0, 255);
 }

 // send ADC data into serial port
 Serial.print("A"); // "A" is header
 for (int i = 0; i < 256; i++) {
  Serial.write(adc_bytes[i]);
 }

 if (time > set_time) {
  set_time = set_time + 70; // RT clock is 70 ms
 }
}
```

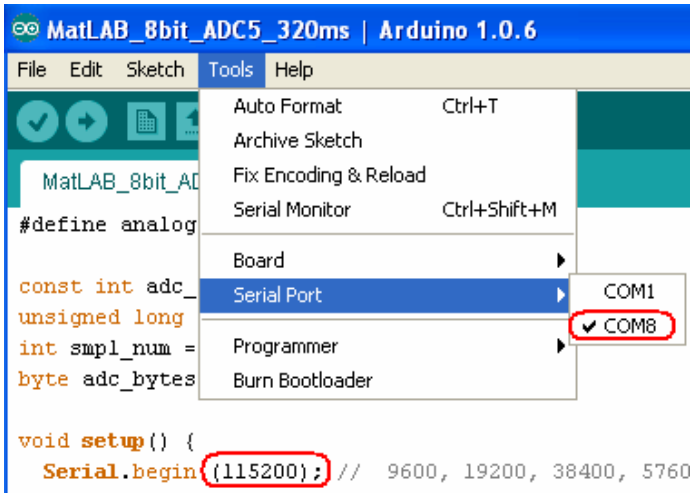

Рис. 7. Определение номера порта а среде Arduino.

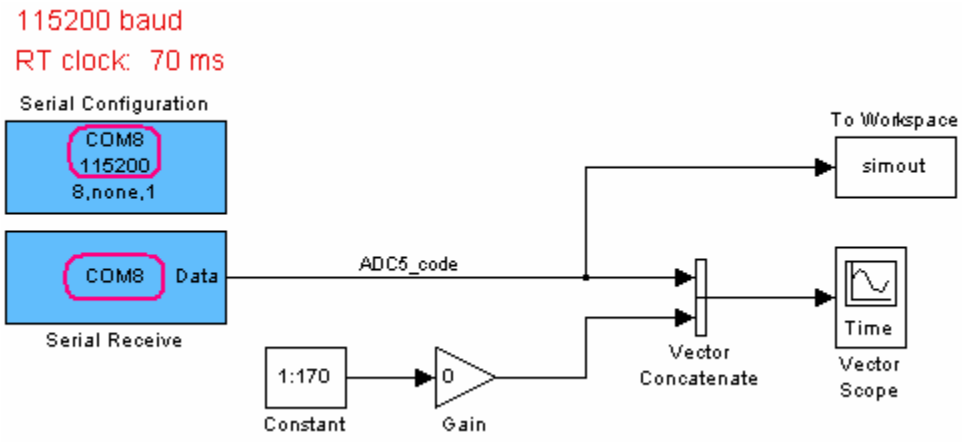

Рис. 8. Simulink модель для приёма АЦП данных контроллера, масштабирования вектора данных по времени, отображения данных в реальном времени и сохранения потока данных в памяти workspace.

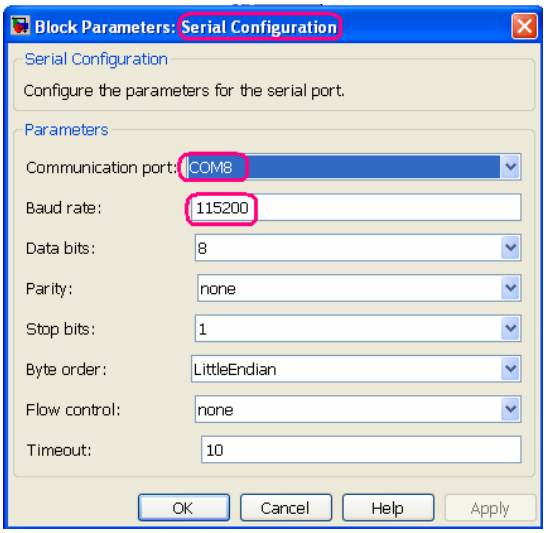

Рис. 9. Параметры COM порта в среде Simulink (блок модели: Serial Configuration)

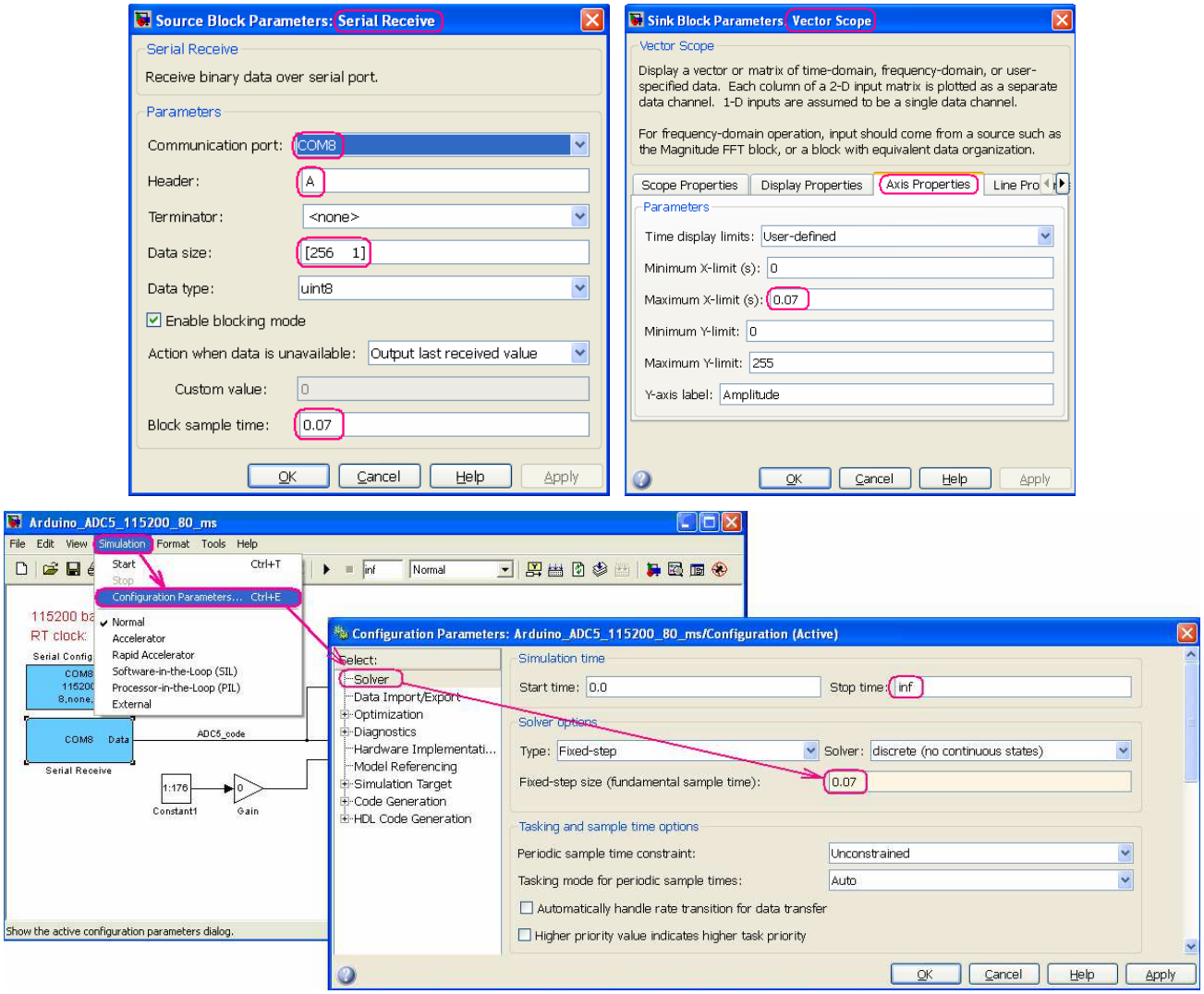

Рис. 10. Параметры блоков Simulink модели и режима моделирования.

Модель запускается нажатием на кнопку Start simulation:

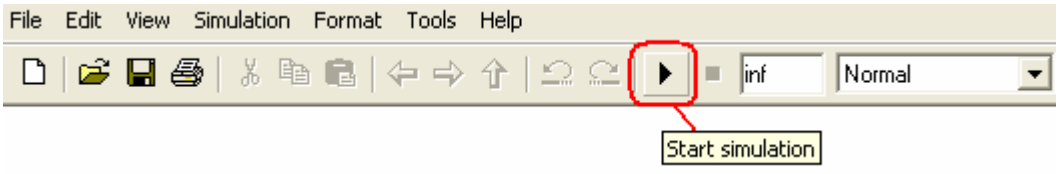

Рис. 11. Кнопка запуска модели.

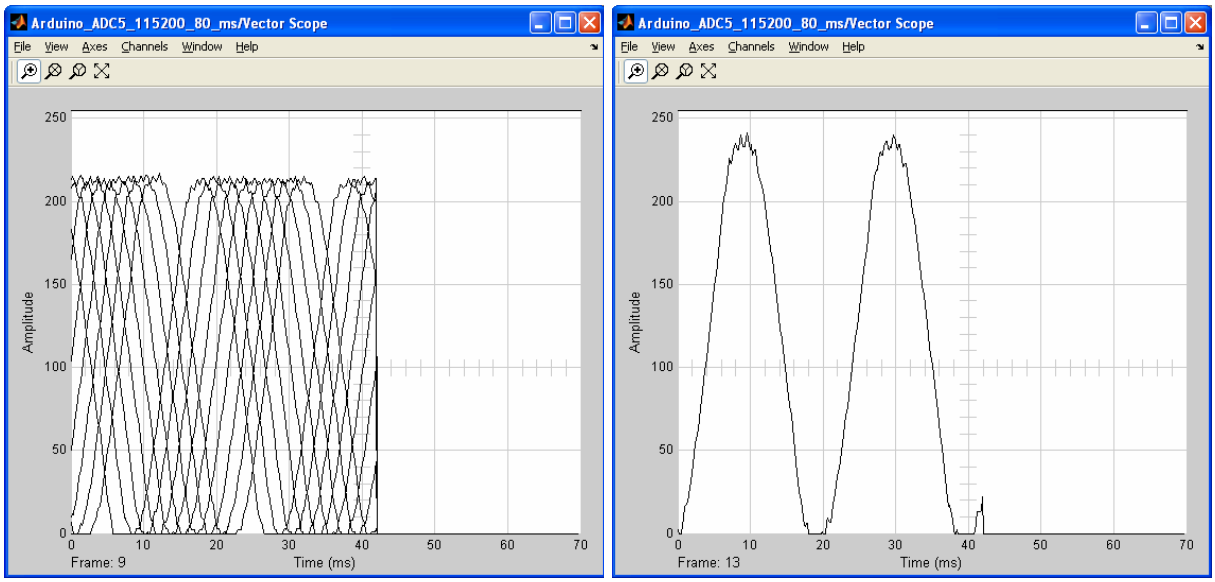

Рис. 12. Вид сетевой наводки (подключение показано на Рис. 6) с наложением кадров (левое окно) и в отдельном кадре (правое окно). Причина смещения сигнала при наложении кадров – отсутствие синхронизации отображения. Примечание: Simulink имеет достаточно средств для построения канала синхронизации.

ПРИМЕРЫ ПОЛУЧЕНИЯ ПРОВЕРЕННЫХ РЕЗУЛЬТАТОВ И ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

Задание 1. Накопление, передача и отображение отмасштабированных данных (см. пример и таблицу времён на стр. 8).

1. Напишите для контроллера Arduino UNO программу циклического считывания показаний АЦП, масштабирования, записи данных в массив 1024 байт и передачи массива в последовательный канал. Программа должна выполняться с максимальной скоростью. Символ A – заголовок передаваемого массива.

Пример программы:

```
const int adc_5 = A5;     // ADC port number
unsigned long set_time;   //  Time of next clock
byte adc_bytes[1024];   // Buffer for ADC data

void setup() {
```

```
  Serial.begin (115200); // bit/s
}

void loop(){
  unsigned long time = millis();   // Current time in ms

  // ADC data capturing
  for (int i = 0; i < 1024; i++) {
    int val = analogRead(adc_5);
    adc_bytes[i] = map(val, 0, 1023, 0, 255);
  }

  // send ADC data into serial port
  Serial.print("A"); // "A" is header
  for (int i = 0; i < 1024; i++) {
    Serial.write(adc_bytes[i]);
  }

  if (time > set_time) {
    set_time = set_time + 270; // RT clock is 270 ms
  }
}
```

2. В среде МатЛАБ составьте программу из Simulink блоков для приема и отображения данных контроллера в реальном времени. Скорость, размер пакета, период принимаемых данных и такт работы модели должны соответствовать соответствующим параметрам контроллера. Отмасштабируйте время отображаемых данных.

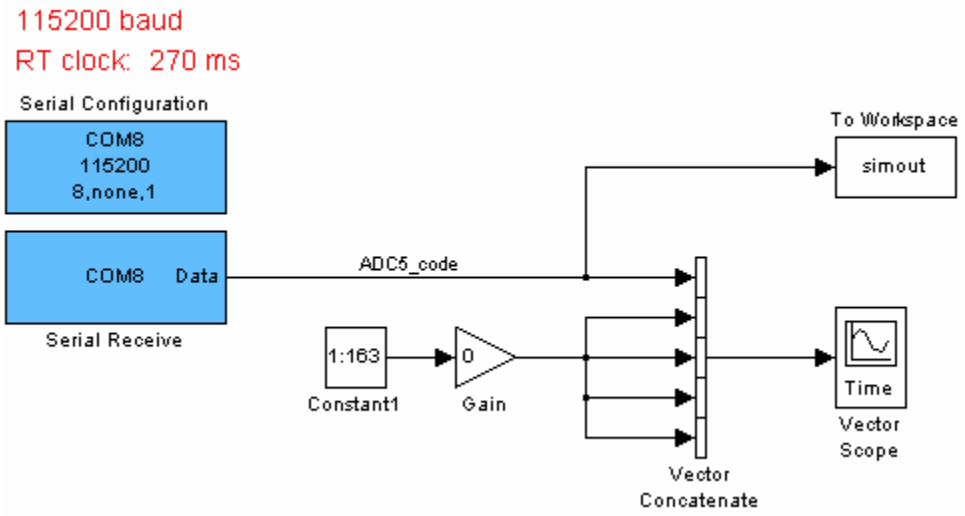

Рис. 13. Simulink модель для приема данных на максимальной частоте: 115200 бод. Объединение векторов (Vector Concatenate) используется для масштабирования сигнала по шкале времени кадра (frame).

3. Проверьте качество канала “Вход АЦП – дисплей МатЛАБ”, например по периоду сетевой 50 Гц наводки на входе АЦП. Для увеличения амплитуды наводки ко входу АЦП подсоедините кусок провода (см. Рис. 6). Амплитуда наводки зависит от расстояния между проводом и вашей рукой.

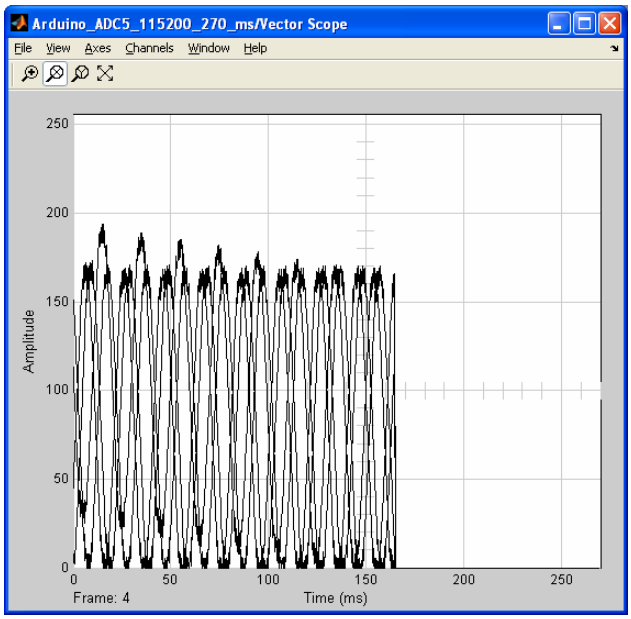

Рис. 14. Наложение 4 кадров при сканировании частоты 50Гц на входе АЦП контроллера Arduino.

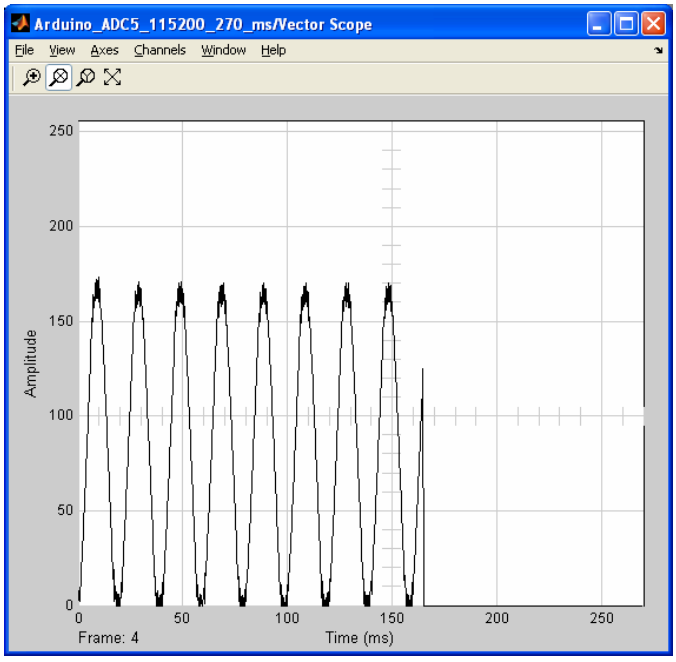

Рис. 15. Частота сети на входе АЦП контроллера, 4 кадр.

Задание 2. Накопление, передача и отображение 10 разрядных данных АЦП.

1. Для контроллера Arduino UNO напишите программу циклического считывания показаний АЦП, записи данных в массив 512 слов и побайтной передачи данных массива в последовательный канал. Программа должна выполняться с максимальной скоростью.

   Пример программы:

```
const int adc_5 = A5;   // ADC port number
unsigned long set_time;  // Time of next clock in ms
word adc_int[512];   // Buffer for ADC data
int val;
byte val_Lo, val_Hi;

void setup() {
  Serial.begin (115200); //  bit/s
}

void loop(){
  unsigned long time = millis();

  // ADC data capturing
  for (int i = 0; i < 512; i++) {
    adc_int[i] = analogRead(adc_5);
  }

  // send ADC data into serial port
  // first low bytes then high bytes
  Serial.print("A"); // "A" is header
  for (int i = 0; i < 512; i++) {
    val = adc_int[i];
    val_Lo = (val << 1) & 0xFE;
    Serial.write(val_Lo);   // Lo byte
  }
  for (int i = 0; i < 512; i++) {
    val = adc_int[i];
    val_Hi = (val >> 6) & 0xE;
    Serial.write(val_Hi);  // Hi byte
  }

  if (time > set_time) {
    set_time = set_time + 160; // RT clock is 160 ms
  }
}
```

2. Составьте программу Simulink для приема восстановления и отображения АЦП данных контроллера. Скорость, размер пакета и период принимаемых данных должны соответствовать соответствующим параметрам контроллера. Отмасштабируйте время отображаемых данных.

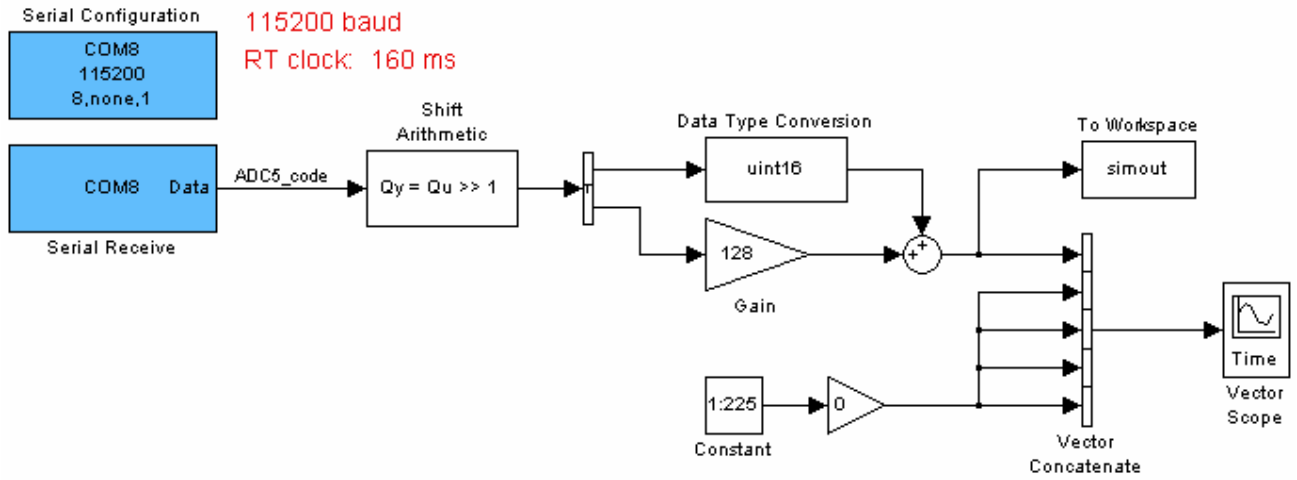

Рис. 16. Программа Simulink для приёма, восстановления и отображения массива данных АЦП контроллера Arduino UNO.

3. Запишите сетевые 50 Гц наводки.

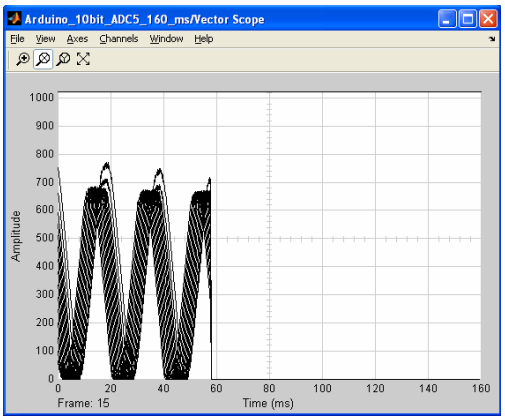

Рис. 17. Наложение 15 кадров при сканировании сетевой наводки 50Гц на входе АЦП контроллера. Период программы: 160 мс. Время заполнения массива данными АЦП: 58 мс. Время передачи 512x2 байт на частоте 115200 бод: 89 мс.

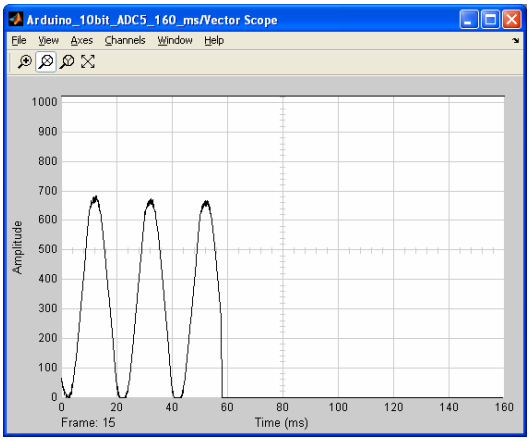

Рис. 18. Последний 15 кадр. Время 3.5 циклов 50 Гц сигнала: 70 мс.

Задание 3. Обработка сигнала m-программой МатЛАБ

1. Сохраните отображаемые в реальном времени данные в workspace памяти МатЛАБ, например, при помощи блока simout (см. Рис. 13).
2. Скопируйте сохраненные данные в рабочий каталог, например:

   save('simout_50Hz','simout');

3. Разработайте m-программу МатЛАБ для отображения архивного АЦП сигнала контроллера.

   Пример кода:

```
clear all
load('simout_50Hz');

d_frame = simout.Time(2)-simout.Time(1);
size_frame = size(simout.Data,1);
sampling = d_frame/(size_frame + 163*4);          % dt
data_size = size(simout.Data,1)*size(simout.Data,2)*size(simout.Data,3);

% time = (0:data_size-1)*sampling;
time = [];
for i = 1:length(simout.Time)
    time = [time (0:1023)*sampling+simout.Time(i)];
end

adc = uint8([]);
for i = 1:size(simout.Data,3)
    adc = [adc simout.Data(:,:,i)'];
end

% frame_num = length(simout.Time) % or size(adc,3) % is 54 frame

if 1 %
    figure
    plot(time, adc, 'b')
    grid on
    xlabel('Time, s');
    ylabel('ADC [0 .. 255], bit');
    title('8 bit ADC frame against Time');
end
```

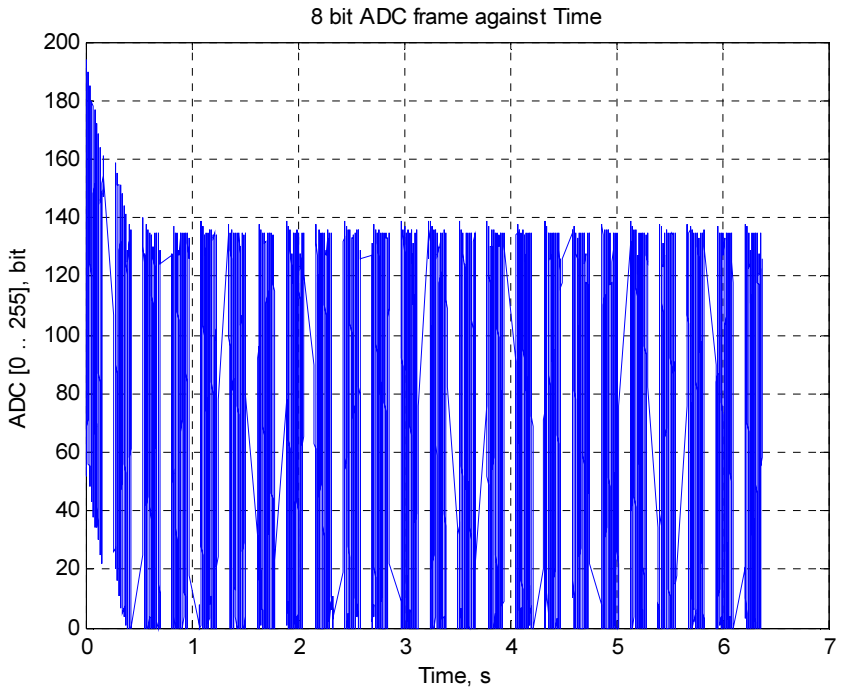

Рис. 19. Покадровое изменение 50 Гц наводки на входе АЦП контроллера Arduino UNO: 24 кадра по 0.27 сек.

4. Разработайте m-программу для вычисления параметров сигнала, например, периода в заданном кадре.

   Пример кода:

```
clear all
load('simout_50Hz');

d_frame = simout.Time(2)-simout.Time(1);
sampling = d_frame/((256 + 176)*4);          % dt
data_size = size(simout.Data,1)*size(simout.Data,2)*size(simout.Data,3); %
<256 x 1 x 54>

%FRAME number
i = 5;
time = (0:1023)*sampling+simout.Time(i);
adc = simout.Data(:,:,i)';
if 1 %
    figure
    plot(time, adc, 'b')
    grid on
    xlabel('Time, s');
    ylabel('ADC [0 .. 255], bit');
    title('8 bit ADC frame against Time');
end
% period
comp_level = 60;
j = 1;
for i = 2:length(adc)
```

```
    if (adc(i) >= comp_level) && (adc(i-1) < comp_level)
        cell_num(j) = i;
        j = j + 1;
    end
end
s_period = diff(time(cell_num));
```

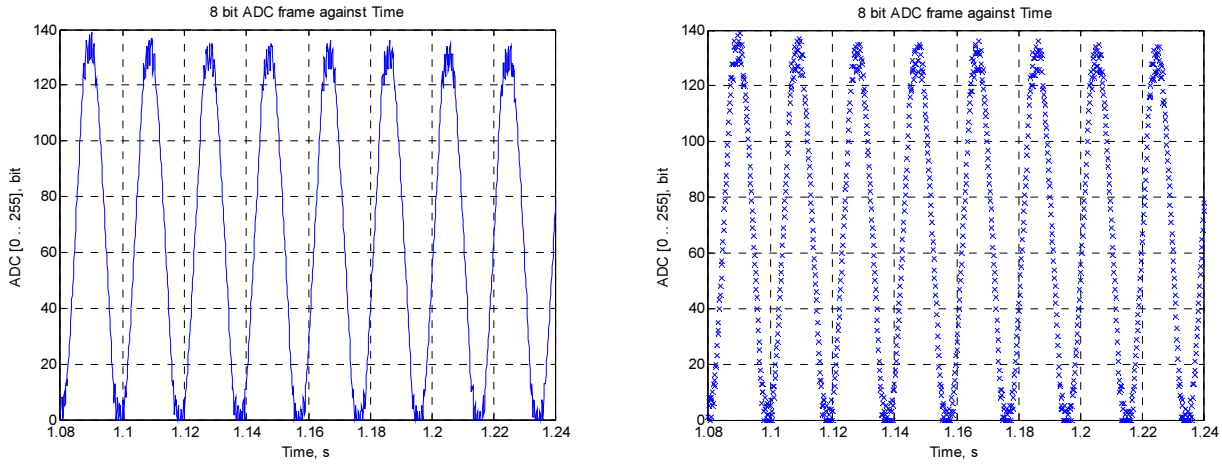

Рис. 20. Непрерывное и поточечное изменение сигнала в выбранном кадре. Время 5 кадра: 1.08 .. 1.24 сек. Размер вектора: 1024 байт. Следовательно время одного считывания и масштабирования сигнала АЦП: 0.156 мс.

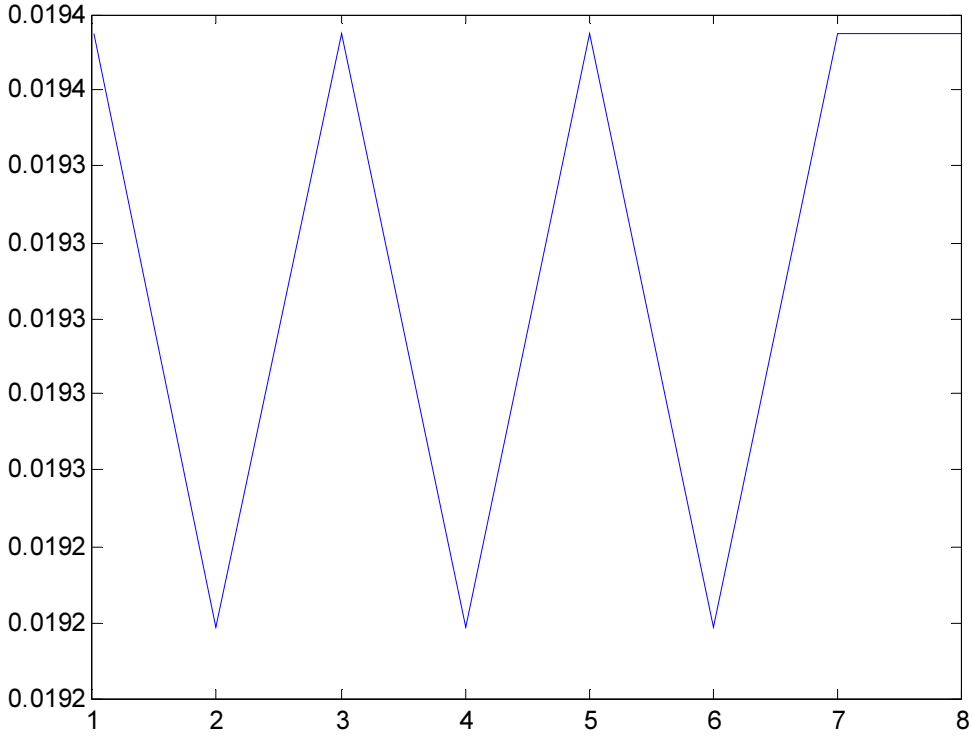

Рис. 21. Период наводки сети 5 кадра: 19.2 .. 19.4 мсек.

Задание 4. Построение спектра сигнала в реальном времени.

1. Для наблюдения частотного спектра сигнала подключите к отображаемому сигналу модели блок быстрого преобразования Фурье (Spectrum Scope: FFT) из раздела библиотеки Simulink > DSP System Toolbox > Sinks.

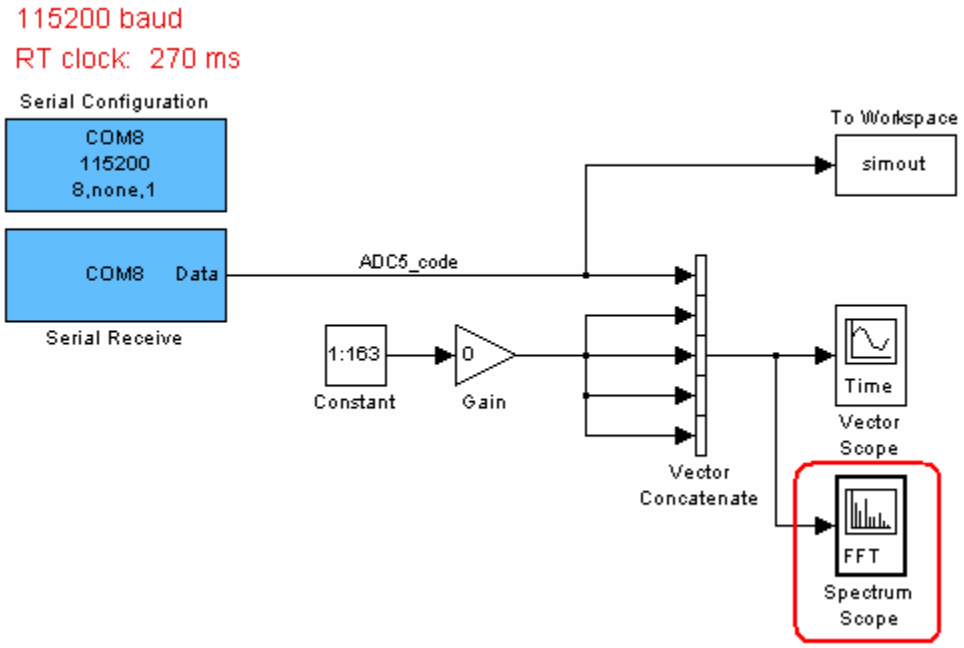

Рис. 22. Модель со спектроскопом.

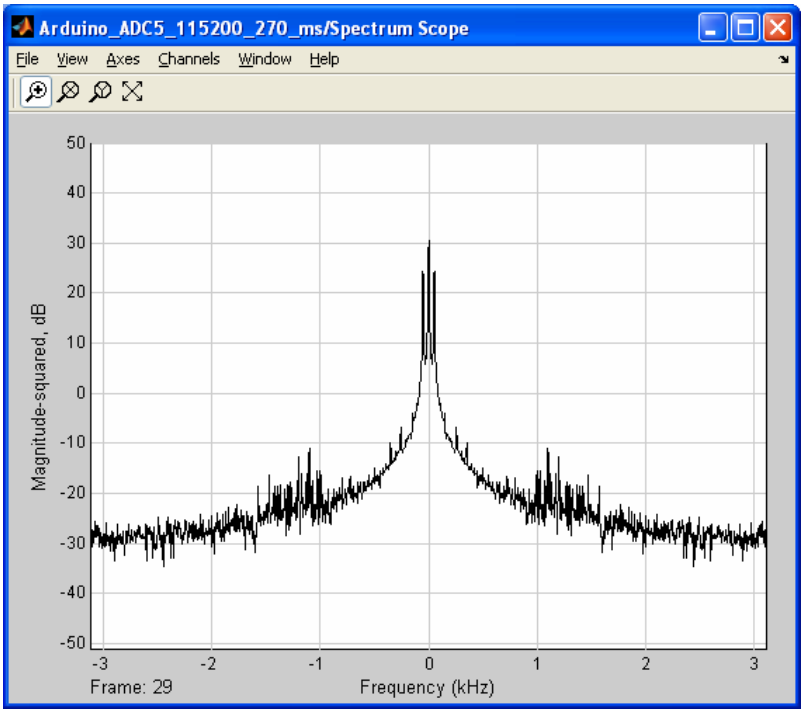

Рис. 23. Спектр сетевой наводки. Сигнал кадра включает 1024 амплитуды и 163x4 нулевых значений.

2. Выделите основную гармонику сигнала: 50 Гц.

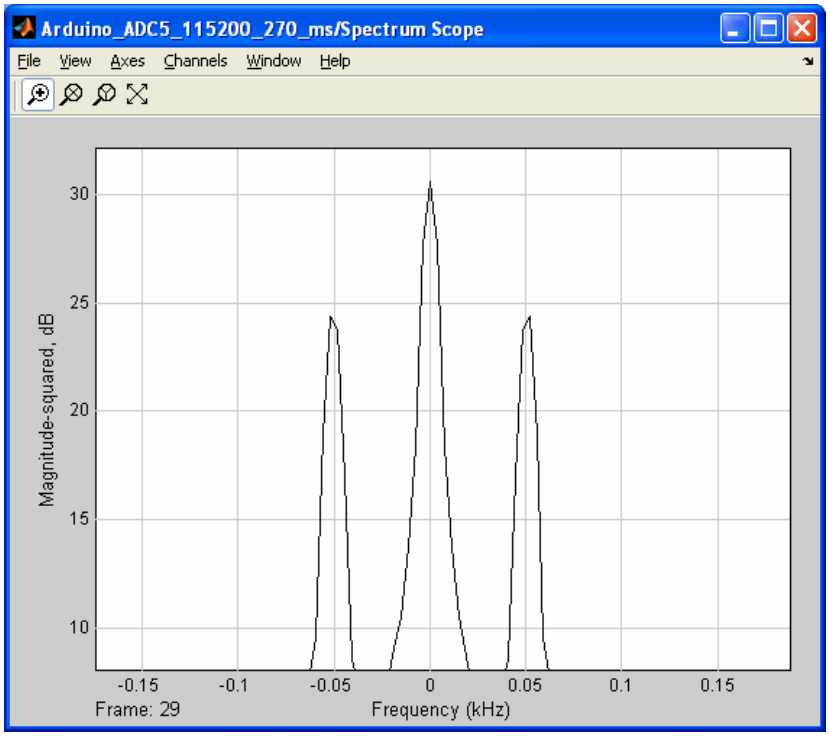

Рис. 24. Гармоника сигнала на частоте 50 Гц.

3. Подключите блок Spectrum Scope: FFT к неотмасштабированному (по времени) сигналу.

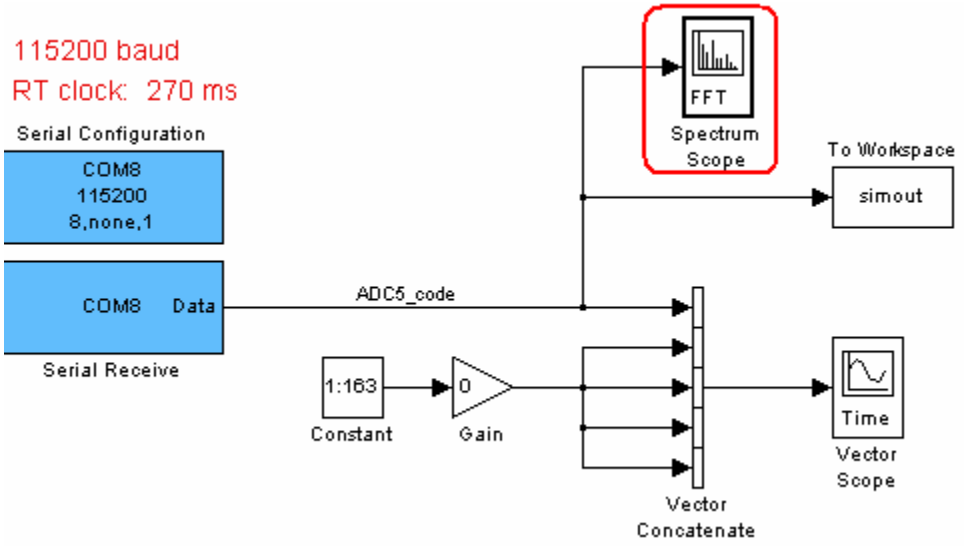

Рис. 25. Перенос точки подключения спектрографа. На входе неотмасштабированный сигнал с меньшей зоной нулевых значений в конце массива (вектора).

4. Настройте блок. Выберите тип отображаемого спектра: Spectrum Type.

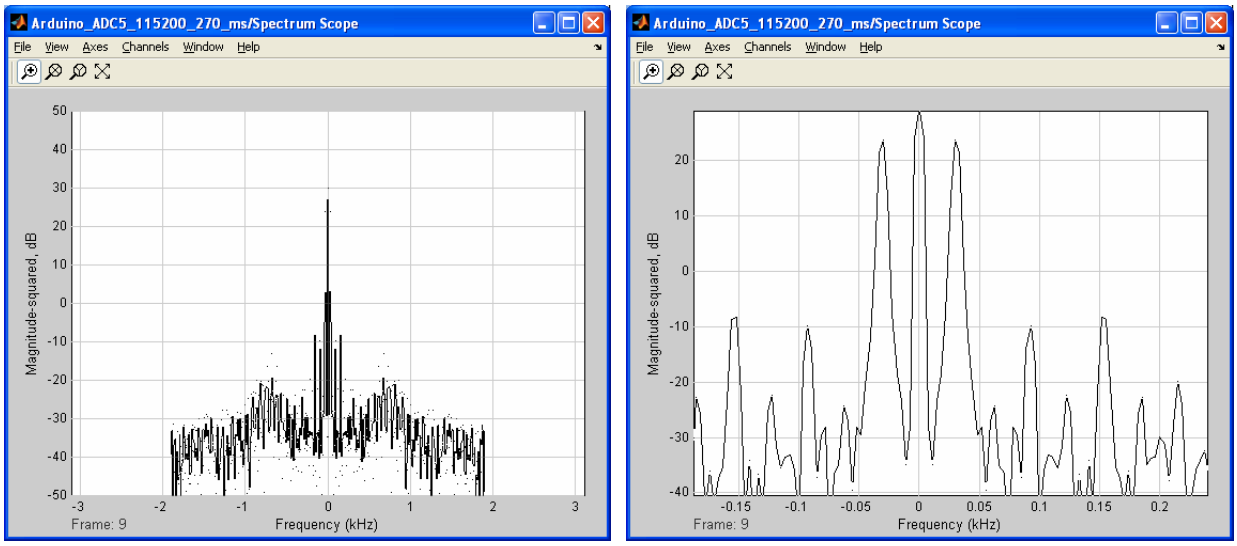

Рис. 26. Параметры спектрометра неотмасштабированного сигнала из 1024 амплитуд.

Задание 5. Построение канала скоростной потоковой передачи и обработки 8р данных в реальном времени без пропуска данных.

1. Напишите для контроллера Arduino UNO программу циклического считывания показаний АЦП, масштабирования и передачи в последовательный канал 2048 байт с заголовком. Программа должна считывать показания АЦП на постоянной частоте без перерывов.

   Пример программы:

```
const int adc_5 = A5;     // ADC port number


void setup() {
  Serial.begin (115200); // bit/s
}


void loop(){
  for (int i = 0; i < 2048; i++) {
    if (i == 0) Serial.print("A "); // "A" is header
    int val = analogRead(adc_5);
    byte adc_byte = map(val, 0, 1023, 0, 255);
    Serial.write(adc_byte);
  }
}
```

2. Настройте модель Simulink (МатЛАБ) на прием данных контроллера.

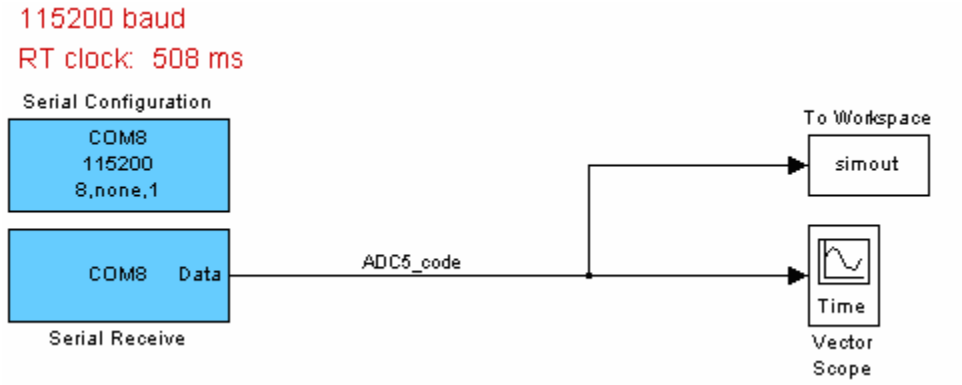

Рис. 27. Пример модели для отображения непрерывного потока данных. Кадр содержит 2048 байт.

3. Настройте время моделирования модели (Меню > Simulation > Configuration Parameters > Solver > Fixed-step size) и такт блока Serial Receive > Block Sample Time, (см. Рис. 10) по периоду 50 Гц сети.

Расчетное время кадра по данным Таблица 1: 254 мс (для 1024 байт) => 508 мс для 2048 байт, В действительности, время кадра программы (в которой считывание АЦП и передача выполняются поочередно) составляет 375 мс.

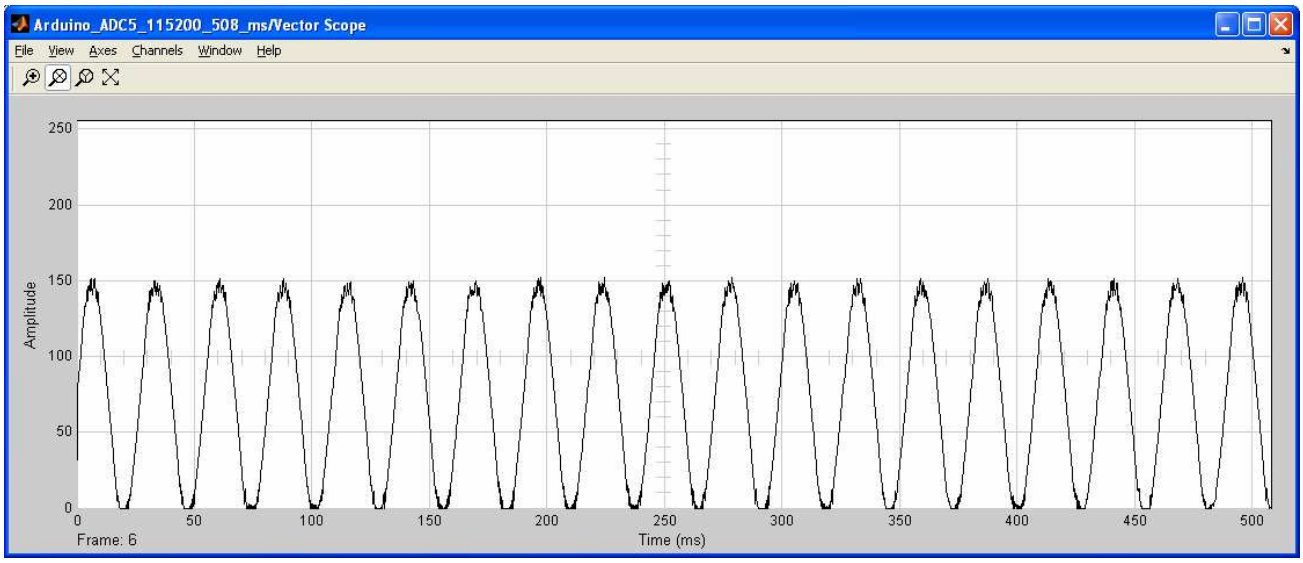

Рис. 28. Кадр графопостроителя Vector Scope. В кадре 18.75 периодов 50 Гц волны. Следовательно, время кадра должно быть 375 мс, а период преобразования АЦП, масштабирования и передачи данных: 0.1831 мс.

4. В командном окне МатЛАБ наберите команду формирования 5 кадрового сигнала.

```
sgnl = [simout.Data(:,1,1)' simout.Data(:,1,2)' simout.Data(:,1,3)' simout.Data(:,1,4)' simout.Data(:,1,5)'];
```

5. Постройте график 5 первых кадров сигнала.

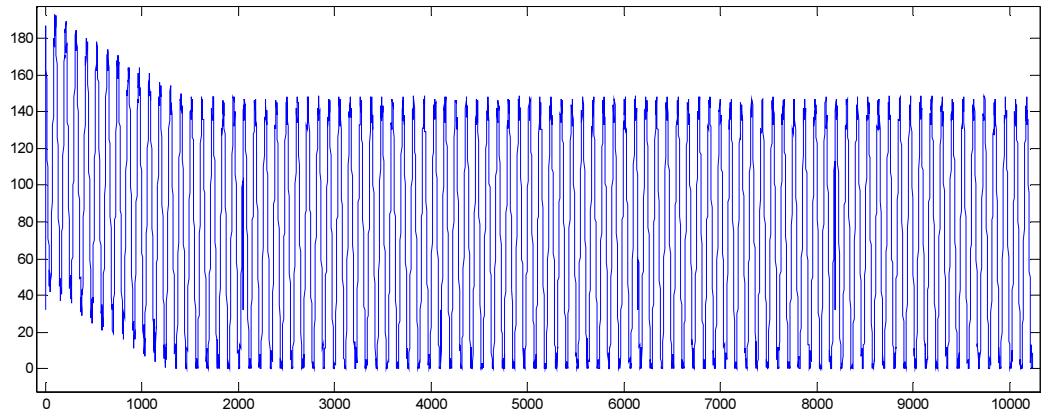

Рис. 29. Пять кадров входного сигнала модели.

6. Рассмотрите качество стыков кадров.

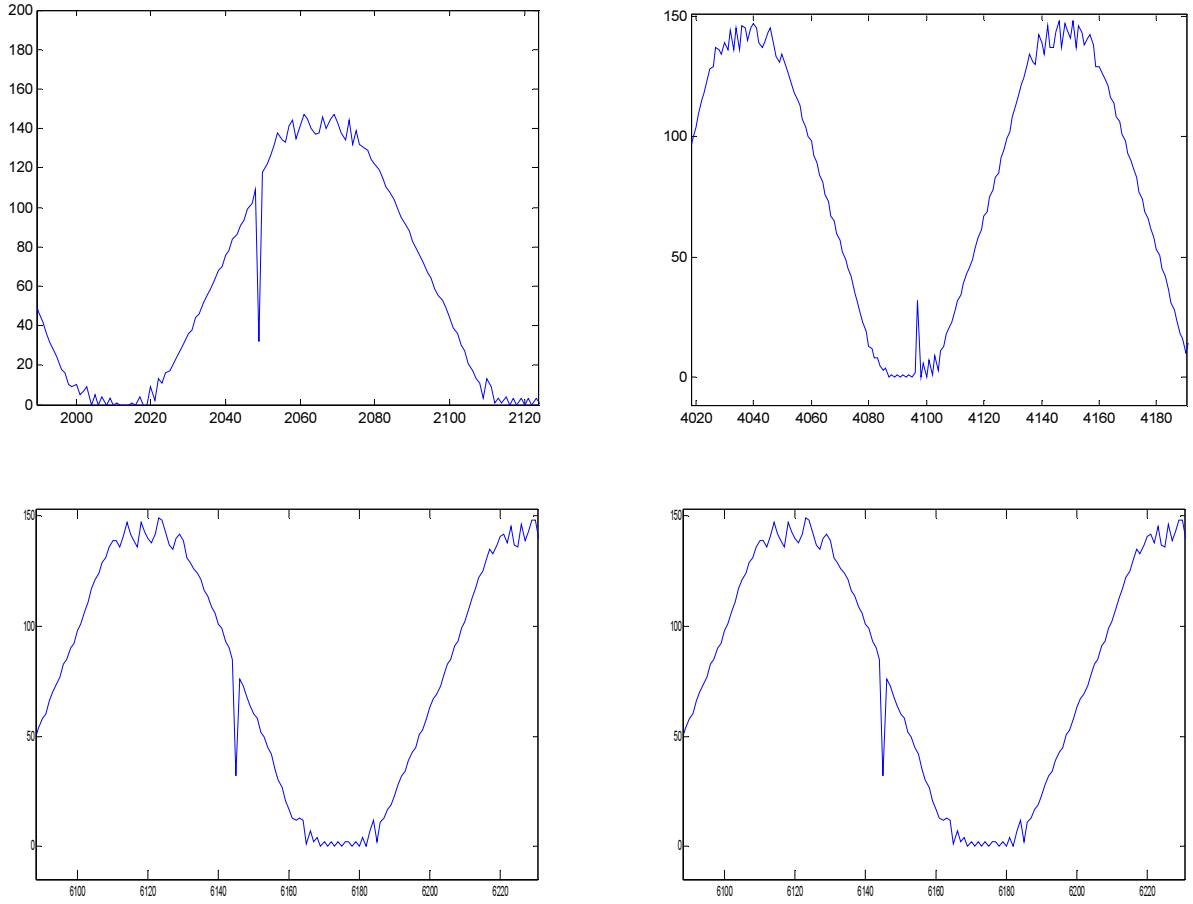

Рис. 30. Стыки пяти кадров. Имеются заметные искажения в первом байте каждого кадра. Заменой первых байт средними значениями между ближайшими точками можно существенно снизить искажения.

7. Подключите ко входному сигналу модели анализатор спектра. Наблюдайте спектр сигнала в реальном времени.

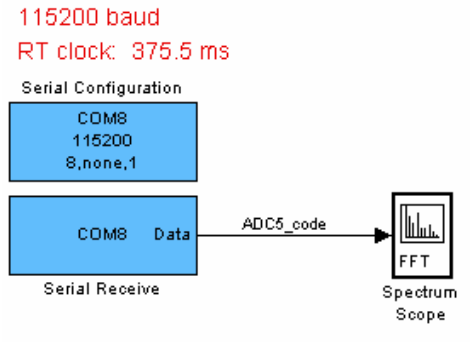

Рис. 31. Модель отображения спектра входного сигнала (АЦП Arduino UNO) в реальном времени.

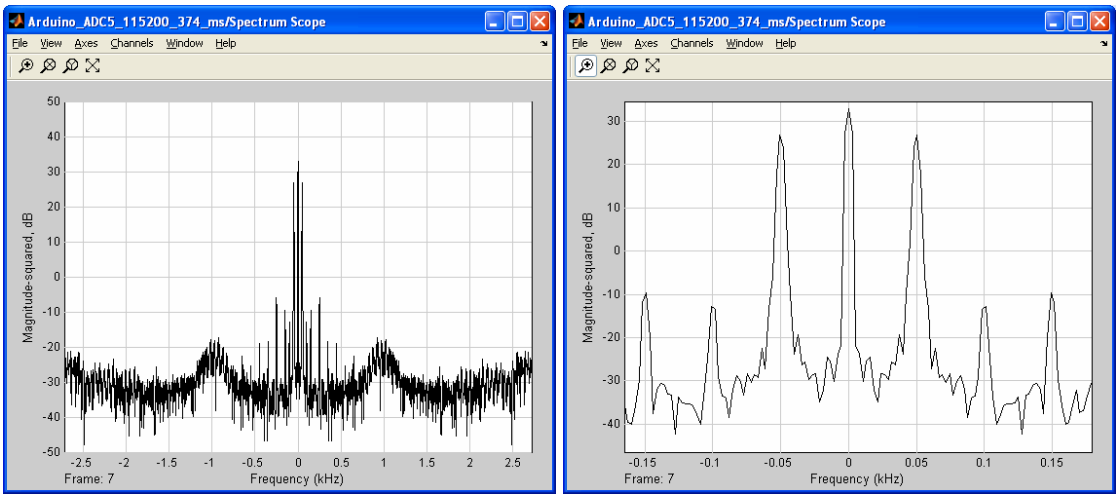

Рис. 32. Спектр сетевой наводки на входе АЦП контроллера Arduino.

8. Подключите ко входному сигналу модели осциллограф Time Scope из библиотеки Simulink > DSP System Toolbox > Sinks.

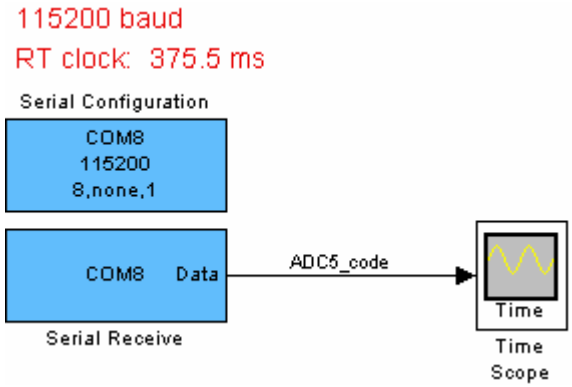

Рис. 33. Осциллограф в модели для отображения входного сигнала контроллера Arduino.

9. Настройте осциллограф на отображение содержимого текущего кадра и частоты сигнала.

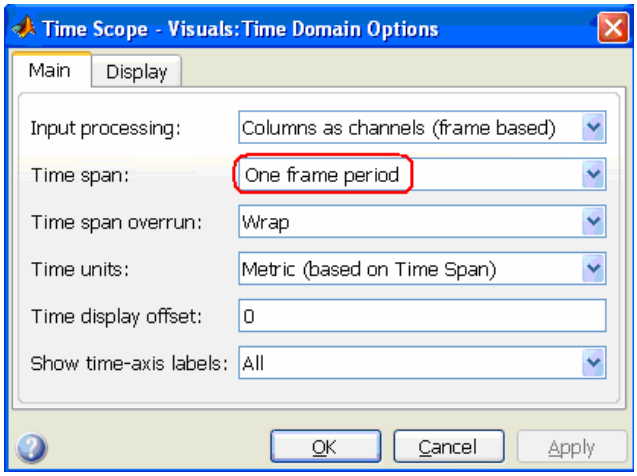

Рис. 34. Настройка осциллографа Time Scope > Menu > View > Properties.

10. Запустите модель и наблюдайте стабильность сигнальных параметров. Изменяя настройки “Time Scope” убедитесь в том, что предлагаемый вариант построения канала отображения сигнала в реальном времени позволяет изменять (настраивать) параметры модели во время её работы .

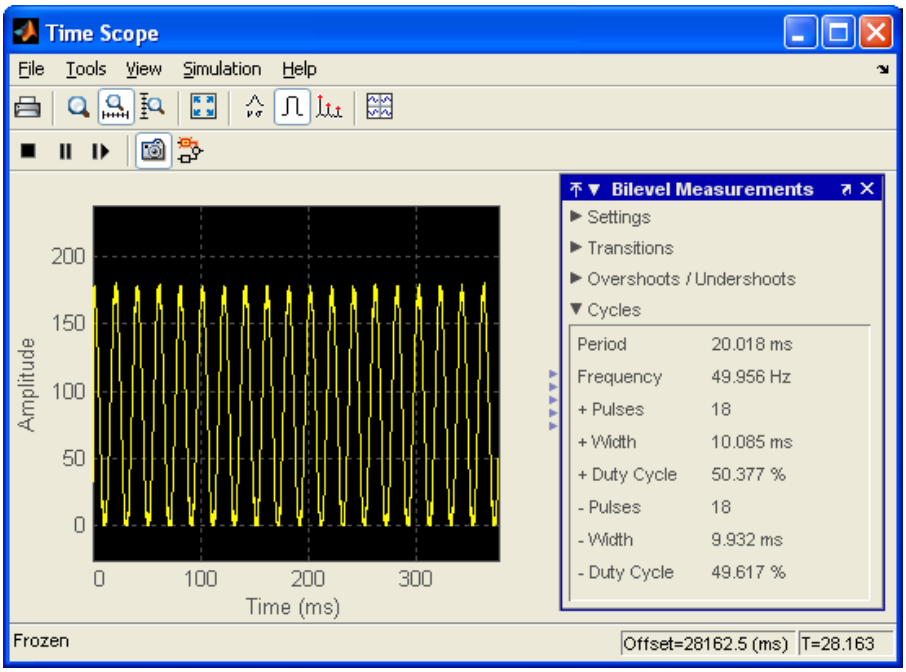

Рис. 35. Отображение сигнала и его параметров в реальном времени на осциллографе Simulink модели.

Последний вариант канала контроллер Arduino – МатЛАБ в сравнении с предыдущими вариантами имеет следующие преимущества.

- не используется память контроллера для накопления АЦП данных;
- обеспечивается малый такт преобразования АЦП с масштабированием, который чуть больше такта преобразования АЦП с масштабированием при отсутствии передачи;

- не требуется масштабирование сигнала по времени в Simulink модели;
- модель содержит меньше блоков;
- практически не ограничен размер вектора и время кадра.

Задание 6. Увеличение частоты сэмплирования сигнала АЦП.

Частоту сэмплирования АЦП контроллера Arduino можно повысить до 15 кГц в 10-разрядном режиме и до 77 кГц в 8-битном режиме [3], заменив библиотечные функции на более быстрый вариант использования регистров микроконтроллера.

Функцию пользователя можно создать в программе *.ino или в системном файле контроллера

...\arduino-1.0.6\hardware\arduino\cores\arduino\ wiring_analog.c, зарегистрировав её в ...\arduino-1.0.6\hardware\arduino\cores\arduino\ Arduino.h

Для построения 8-разрядного высокоскоростного канала Arduino – МатЛАБ необходимо выполнить следующее.

1. Напишите программу определения времени заполнения массива АЦП данными с отображением результата в окне 

. Размер массива должен быть достаточно большим, например, в половину памяти SRAM. Для увеличения точности следует измерять время многократного заполнения массива.

   Пример программы:

```
byte adc_bytes[1024];   // Резервирование массива для АЦП данных

void setup() {
  Serial.begin (57600); // bit/s

  ADCSRA = (1 << ADEN) // Включение АЦП
  |(1 << ADPS2);      // Установка предделителя преобразователя на 8
  ADMUX = (1<< ADLAR) | (1 << REFS0) // Подключение внешнего ИОН
  |(1 << MUX2) |(0 << MUX1) |(1 << MUX0);  // подключение АЦП  A5 == 101
}

void loop(){
  unsigned long time_start = millis();
  for (int j = 0; j < 100; j++) {
    for (int i = 0; i < 1024; i++) {
      ADCSRA |= (1 << ADSC); //  Запуск преобразования АЦП
      while ((ADCSRA & (1 << ADIF)) == 0);//Ожидание флага окончания
преобразования
      adc_bytes[i] = ADCH; // Считывание полученного значения
    }
  }
```

```
  unsigned long time_end = millis();
  unsigned int dt = time_end - time_start;
  Serial.println (dt);
}
```

Сто заполнений массива из 1024 байт выполнено за 1542 мс.

2. Дополните однократное заполнение массива данными АЦП последующей передачей всего массива в последовательный порт на максимальной скорости.

   Пример программы:

```
byte adc_bytes[1024];   // Резервирование массива для АЦП данных

void setup() {
  Serial.begin (115200); // bit/s
  ADCSRA = (1 << ADEN) // Включение АЦП
  |(1 << ADPS2); // Установка предделителя преобразователя на 8
  ADMUX = (1<< ADLAR) | (1 << REFS0) // Подключение внешнего ИОН
  |(1 << MUX2) |(0 << MUX1) |(1 << MUX0);  // подключение АЦП  A5 == 101
}

void loop(){
  for (int i = 0; i < 1024; i++) {
    ADCSRA |= (1 << ADSC); //  Запуск преобразования АЦП
    while ((ADCSRA & (1 << ADIF)) == 0); //Ожидание флага окончания преобразования
    adc_bytes[i] = ADCH; // Считываем полученное значение
  }
  // send ADC data into serial port
  Serial.print("A"); // "A" is header
  for (int i = 0; i < 1024; i++) {
    Serial.write(adc_bytes[i]);
  }
}
```

3. В модели Simulink (Рис. 36) в формате 0.01542 пропишите экспериментальное значение времени записи в массив, а именно, в строке “Block sample time” блока “Serial Receive” и в строке меню > simulation > Configuration Parameters > Fixed-step size (fundamental sample time).

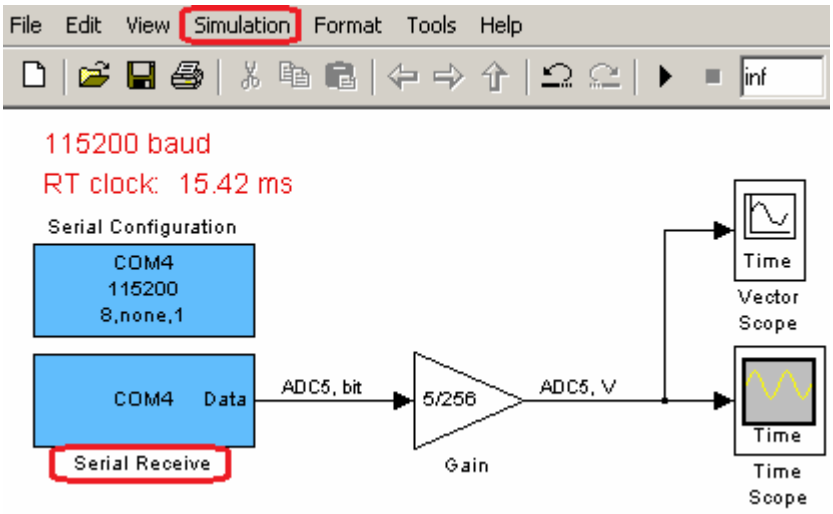

Рис. 36. Модель Simulink для приема и отображения данных из COM порта.

4. Подключите на вход АЦП тестовый сигнал. Запустите программу Arduino и, затем, модель Simulink (MatLAB). Сравните известные параметры сигнала с параметрами наблюдаемого сигнала. Проверить работу тракта можно по отображению поочередно подключаемых выходных напряжений платы Arduino: 0В; 3.3В и 5В.

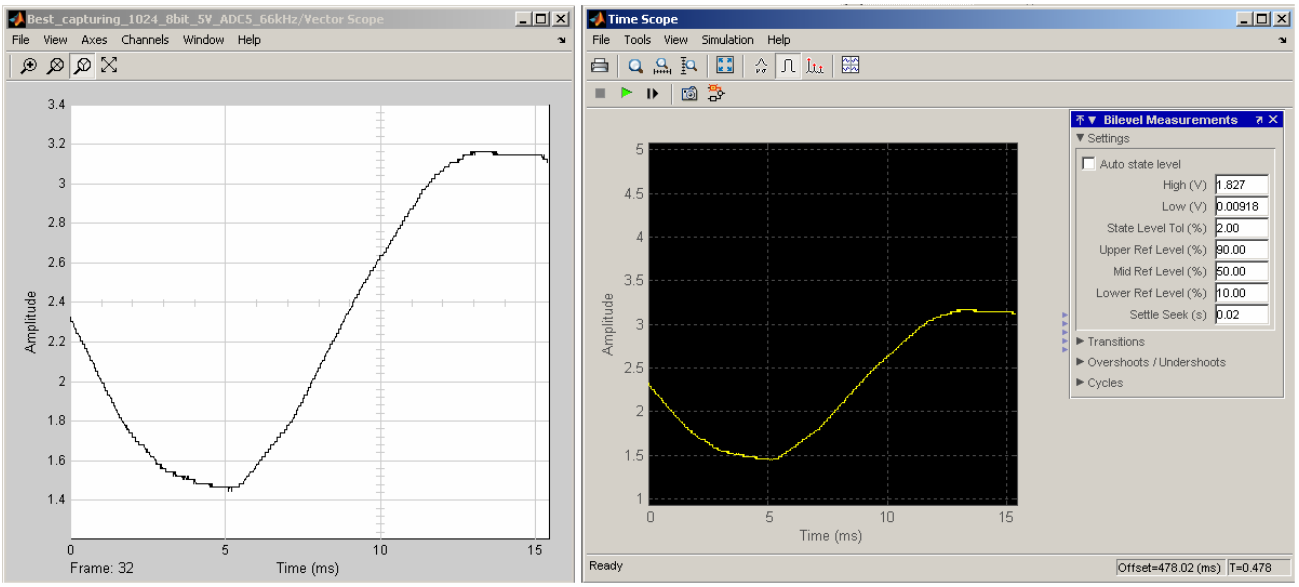

Рис. 37. Отображение в реальном времени сетевой 50 Гц наводки. Кадр включает 1024 точки. Время кадра 15.42 мсек. Частота сэмплирования 66 КГц (как 1/(0.01542_сек/1024)). Отображаемый сигнал имеет разрывы: процесс записи прерывается передачей кадра в последовательный канал.

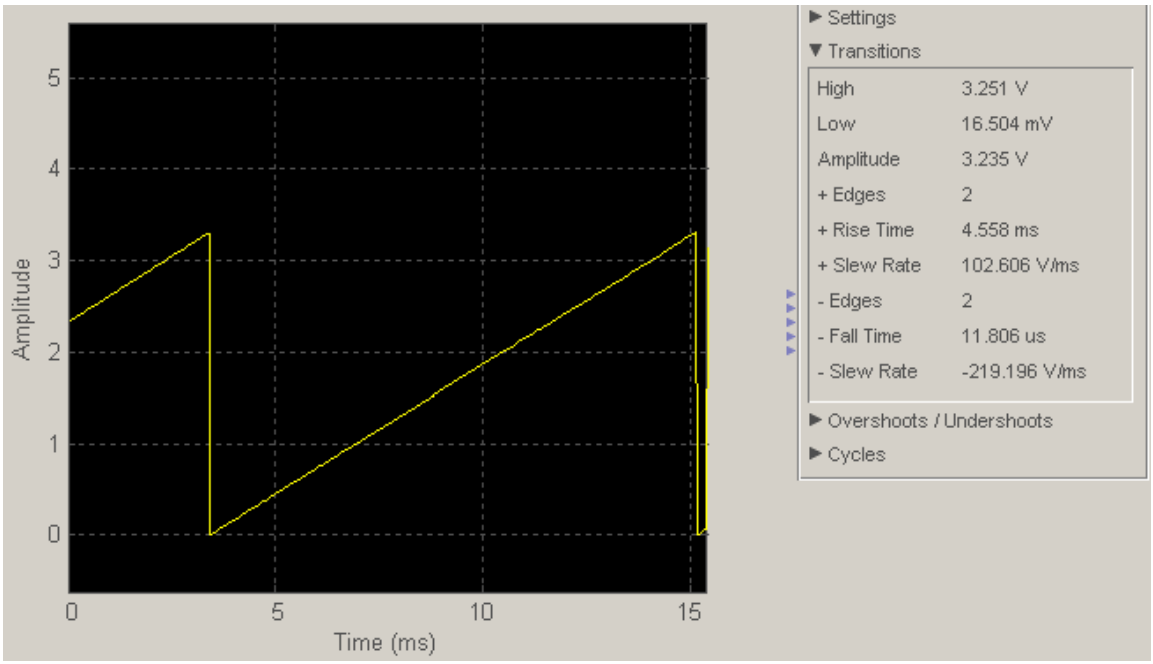

Рис. 38. Отображения в режиме реального времени пилообразного сигнала 0 .. 3.3 В, сформированного на контроллере Teensy 3.1 с 12 разрядным ЦАП и подключенного к шестому АЦП (A5) контроллера Arduino.

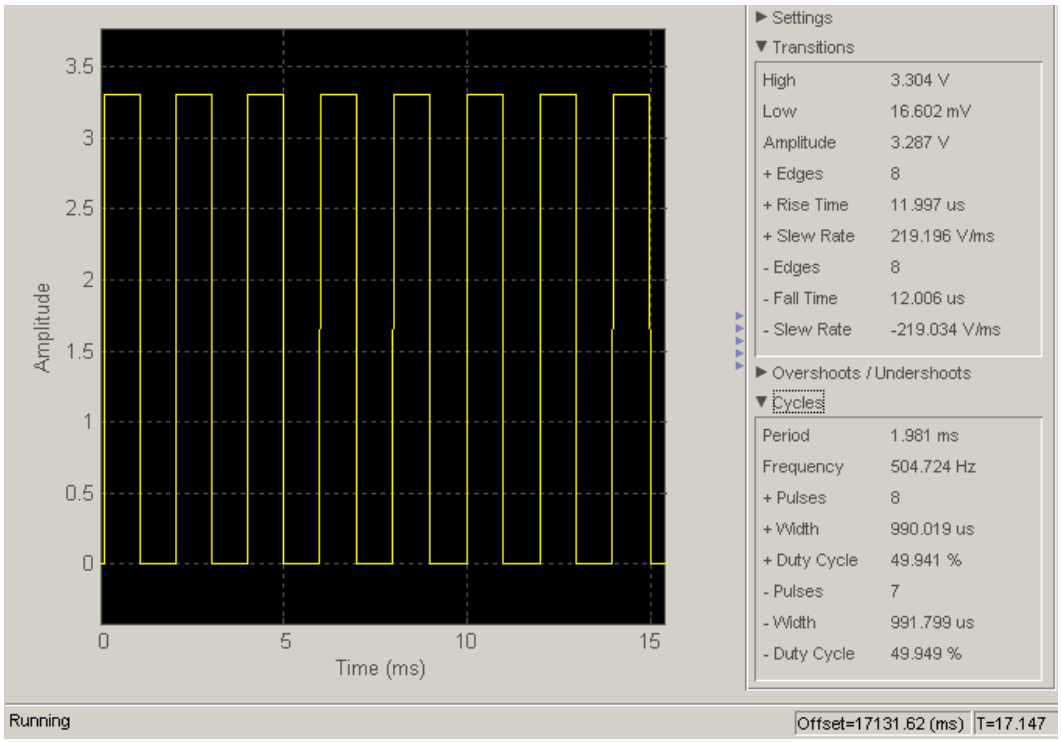

Рис. 39. Сигнал 500 Гц контроллера Teensy 3.1. Ошибка такта (15.42 мсек) Simulink модели без дополнительной корректировки меньше 1% (как 100%*(504.72Гц - 500Гц)/500Гц). Погрешность можно уменьшить корректировкой RT такта, как показано в п.3 этого задания.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Сравните периоды преобразования АЦП первого и последнего заданий.
2. Почему для построения спектра сигнала рекомендуется брать выборку размером кратную двум?
3. Какова задержка потоковой передачи 1024 байт на частоте 115200 бит/с при следующих параметрах передачи?
    - Data bits: 8
    - Parity: none
    - Stop bits: 1
    - Flow control: none

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. DSP System Toolbox. http://matlab.ru/datasheets/Dsp-system-toolbox-Ru.pdf
2. Dr. Bob Davidov. Система термостатирования на базе USB интерфейса Lcard E14-440 (S-function) http://portalnp.ru/2013/09/1036
3. ATmega48A/PA/88A/PA/168A/PA/328/P. ATMEL 8-bit microcontroller with 4/8/16/32kbytes in-system programmable flash DATASHEET (страница 237) http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf
4. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov.