Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Специальное машиностроение»

КАФЕДРА «Колесные машины»

Отчёт о выполнении лабораторной работы №3

по курсу

«Управление техническими системами»

на тему

«Изучение принципов работы шагового двигателя и принципа программного управления для систем автоматического управления»

Студент СМ10-71 _____ В.Б. Сухоносенко
(подпись, дата) (Ф.И.О.)

Преподаватель _____ А.А. Смирнов
(подпись, дата) (Ф.И.О.)

2025 г.

Содержание

1 Исходные данные

Задачей лабораторной работы №3 являлось:

1. Изучение устройства и принципов работы шагового двигателя и его применение в автомобилях;

2. Изучение платы Arduino и основ ее программирования;

3. Изучение лабораторной установки. Вычисление в MathCAD необходимых параметров в соответствии с индивидуальным вариантом;

4. Написание управляющей программы для микроконтроллера;

5. Загрузка программы в микроконтроллер и ее тестирование;

6. Запись в файл с помощью модуля АЦП управляющей последовательности, подающейся на обмотку шагового двигателя;

7. Обработка полученного сигнала (MATLAB) с целью определения числа импульсов на прямом и обратном ходе, длительности этих импульсов, а также полного времени прямого и обратного хода.

На рис. 1.1, рис. 1.2 показаны схема лабораторной установки, и схема, по которой должны подаваться сигналы на обмотки шагового двигателя.

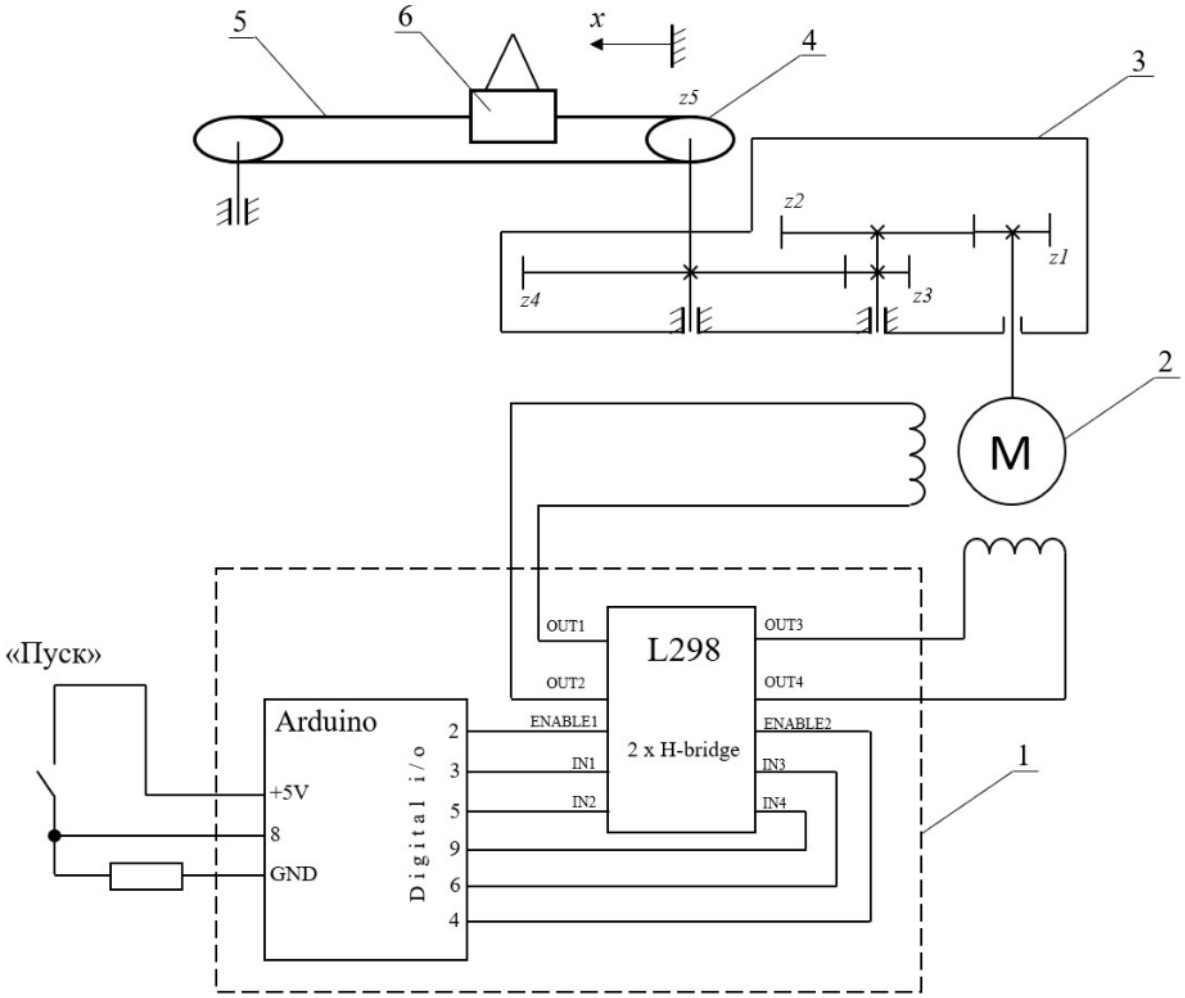
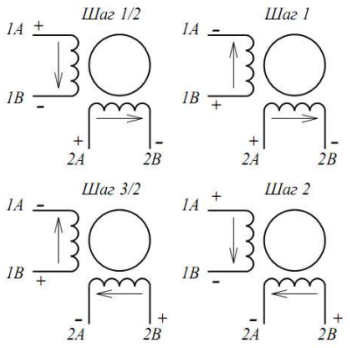


Рисунок 1.1 — Схема лабораторной установки



Шаг	Контакты обмоток			
	1A	1B	2A	2B
1/2	HIGH	LOW	HIGH	LOW
1	LOW	HIGH	HIGH	LOW
3/2	LOW	HIGH	LOW	HIGH
2	HIGH	LOW	LOW	HIGH

Рисунок 1.2 — Подача сигналов на обмотки шагового двигателя

2 Выполнение работы

2.1 Вычисление параметров установки и закона управления

В приложении MATLAB создан файл с расчётом параметров установки: передаточное число редуктора (см. рис. 1.1) равно $i_r = \frac{z_2 z_4}{z_1 z_3}$. Полное передаточное отношение механизма $i_h = h z_5 / i_r$.

```
format shortG;

z=[16 56 22 88 14]; % числа зубьев шестерен
ir= z(2)*z(4)/(z(1)*z(3)); %Передаточное число редуктора
h=2.03; % Шаг зубчатого ремня,мм
% Полное передаточное отношение - мм/об
ih= h*z(5)/ir;
step = 48;

lab=struct('z',z,'ir',ir,'h',h,'ih',ih,'step',step);
lab
```

```
lab = struct with fields:
z: [16 56 22 88 14]
ir: 14
h: 2.03
ih: 2.03
step: 48
```

Далее в соответствии с вариантом 4 рассчитаны параметры закона управления - скорость движения, время простоя:

```
% Закон управления
% по вариантам:
x1 = [0.068 0.080 0.091 0.078];
t1 = [4.8 5.5 8.5 7.5];
t2 = [6 6.5 9 8.2];
```

```
t3 = [11.7 14.2 16.5 20.1];
Izakon= 4; % Выбран вариант 4
x1=x1(Izakon);
t1=t1(Izakon);
t2=t2(Izakon);
t3=t3(Izakon);

v1 =x1/t1 % Линейная скорость каретки на прямом ходе m/s
```

```
v1 =
0.0104
```

```
v2 = x1/(t3-t2) % Линейная скорость каретки на обратном ходе m/s
```

```
v2 =
0.0065546
```

```
prostoi=(t2-t1)*1e3 % Длительность простоя ms
```

```
prostoi =
700
```

Для проверки построен график закона управления (рис. ??)

```
zakon.time = [0 t1 t2 t3];
zakon.x = [0 x1 x1 0];
plot(zakon.time,1000*zakon.x,'-')
gostplot("Закон управления кареткой",
 "Время,с", "Перемещение, мм",width=2)
grid off
ylim([0 x1*1e3+5])
yticks([0 x1*1e3])
xline(t1,'--')
```

```
xline(t2,'--')
yline(x1*1e3,'--')
```

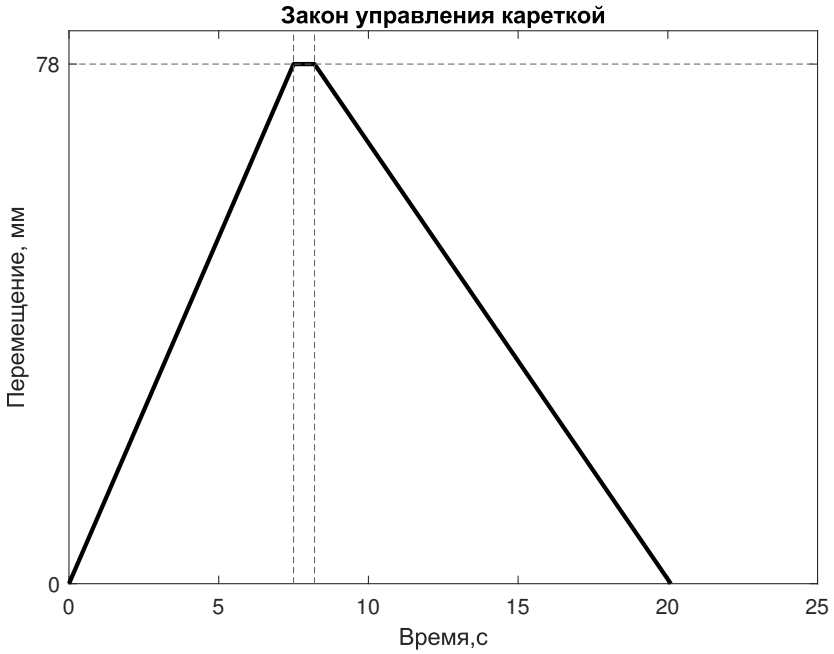


Рисунок 2.1 — Полученный закон управления

Время простоя при движении вперёд и назад stepDelay $= i_h/(2 \cdot \text{step} \cdot v)$ вычисляется из общего времени делить на число шагов в обороте

```
% Время задержки
stepDelay(1)=ih/(2*step*v1);
stepDelay(2)=ih/(2*step*v2); % Время (задержка) шага в миллисекундах
stepDelay_mks = round(stepDelay *1e3,0) % Перевод в микросекунды
```

```
stepDelay_mks = 1x2
2033        3226
```

Общее число шагов - длина всего пути делить на путь, проходимый за один шаг:

```
steps=round(step*x1/ih*1000,0) % Общее число шагов
```

```
steps =
1844
```

2.2 Программа Arduino

Программа состоит из двух файлов - в одном объявляется функция GoSteps, которая в ответ на заданное число шагов, задержку и выбранное направление подаёт сигналы на шаговый двигатель, и основного файла, в котором задаётся, чтобы при нажатии на кнопку каретка перемещалась на расстояние, соответствующее варианту

2.2.1 Файл GoSteps

```
//(c)SUHONOSENKO, 2025
void GoSteps(int nSteps, int nSemiStepDelay, bool dir)
{
int i = nSteps;

digitalWrite(Enable1Pin,HIGH);
digitalWrite(Enable2Pin,HIGH);
while (i>0)
{
// 1/2
digitalWrite(IN1Pin, dir xor HIGH);
digitalWrite(IN2Pin, dir xor LOW);
digitalWrite(IN3Pin, HIGH);
digitalWrite(IN4Pin, LOW);
delayMicroseconds(nSemiStepDelay);
//Step1
digitalWrite(IN1Pin, dir xor LOW);
digitalWrite(IN2Pin, dir xor HIGH);
digitalWrite(IN3Pin, HIGH);
digitalWrite(IN4Pin, LOW);
delayMicroseconds(nSemiStepDelay);
i=i-1;
if (i>0)
{
```

```
//Step 3/2
digitalWrite(IN1Pin, dir xor LOW);
digitalWrite(IN2Pin, dir xor HIGH);
digitalWrite(IN3Pin, LOW);
digitalWrite(IN4Pin, HIGH);
delayMicroseconds(nSemiStepDelay);
//Step 2
digitalWrite(IN1Pin, dir xor HIGH);
digitalWrite(IN2Pin, dir xor LOW);
digitalWrite(IN3Pin, LOW);
digitalWrite(IN4Pin, HIGH);
delayMicroseconds(nSemiStepDelay);
i=i-1;
}
}
digitalWrite(Enable1Pin,LOW);
digitalWrite(Enable2Pin,LOW);
}
```

2.2.2 Файл основной программы

```
//(c)SUHONOSENKO, 2025
void GoSteps(int, int, bool);

const int Enable1Pin =2;
const int IN1Pin = 3;
const int IN2Pin = 5;
const int Enable2Pin=4;
const int IN3Pin=6;
const int IN4Pin = 9;
const int ButtonPin = 8;
bool ButtonState = 1;
```

```
void setup() {
// put your setup code here, to run once:
pinMode(Enable1Pin, OUTPUT);
pinMode(Enable2Pin, OUTPUT);
pinMode(IN1Pin, OUTPUT);
pinMode(IN2Pin, OUTPUT);
pinMode(IN3Pin, OUTPUT);
pinMode(IN4Pin, OUTPUT);
digitalWrite(Enable1Pin,LOW);
digitalWrite(Enable2Pin,LOW);
}

void loop() {
// put your main code here, to run repeatedly:
ButtonState=digitalRead(ButtonPin);
if (ButtonState)
{
GoSteps(1844,2033,0);
delay(700);
GoSteps(1844,3226,1);
}
}
```

2.3 Проведение эксперимента и обработка данных

Проект Arduino загружен на микроконтроллер, проведена запись значений после нажатия кнопки при движении каретки. Замечено, что в обратном направлении слышно потрескивание - скорее всего, шаговый двигатель пропускал шаги. С помощью АЦП и программы LGraph сняты значения напряжения с одной из обмоток двигателя.

Ниже приведён код для вывода результатов: предварительно txt-файл импортирован в рабочее пространство MATLAB в виде структуры (exp.time,exp.V)

```
%Вывод результатов эксперимента:
load("experiment_data.mat")

figure
tiledlayout(2,2,'TileSpacing','compact','Padding','compact')
nexttile(1,1:2)
plot(exp.time,exp.V)
ylim([-0.2 5.2])
xlim([0 exp.time(end)])
gostplot("Напряжение с АЦП","Время,с","Напряжение, В",keep_color=true)

nexttile()
plot(exp.time,exp.V,'-x')
hold on
gostplot("Приближено(вперёд)","Время,с","Напряжение, В",keep_color=true)
grid on
xlim([t1/2 t1/2+0.02])
ylim([-0.2 5.2])

nexttile()
plot(exp.time,exp.V,'-x')
hold on
gostplot("Приближено(назад)","Время,с","",keep_color=true)
grid on

xlim([t2+(t3-t2)/2 t2+(t3-t2)/2+0.02])
ylim([-0.2 5.2])
yticklabels([])
```

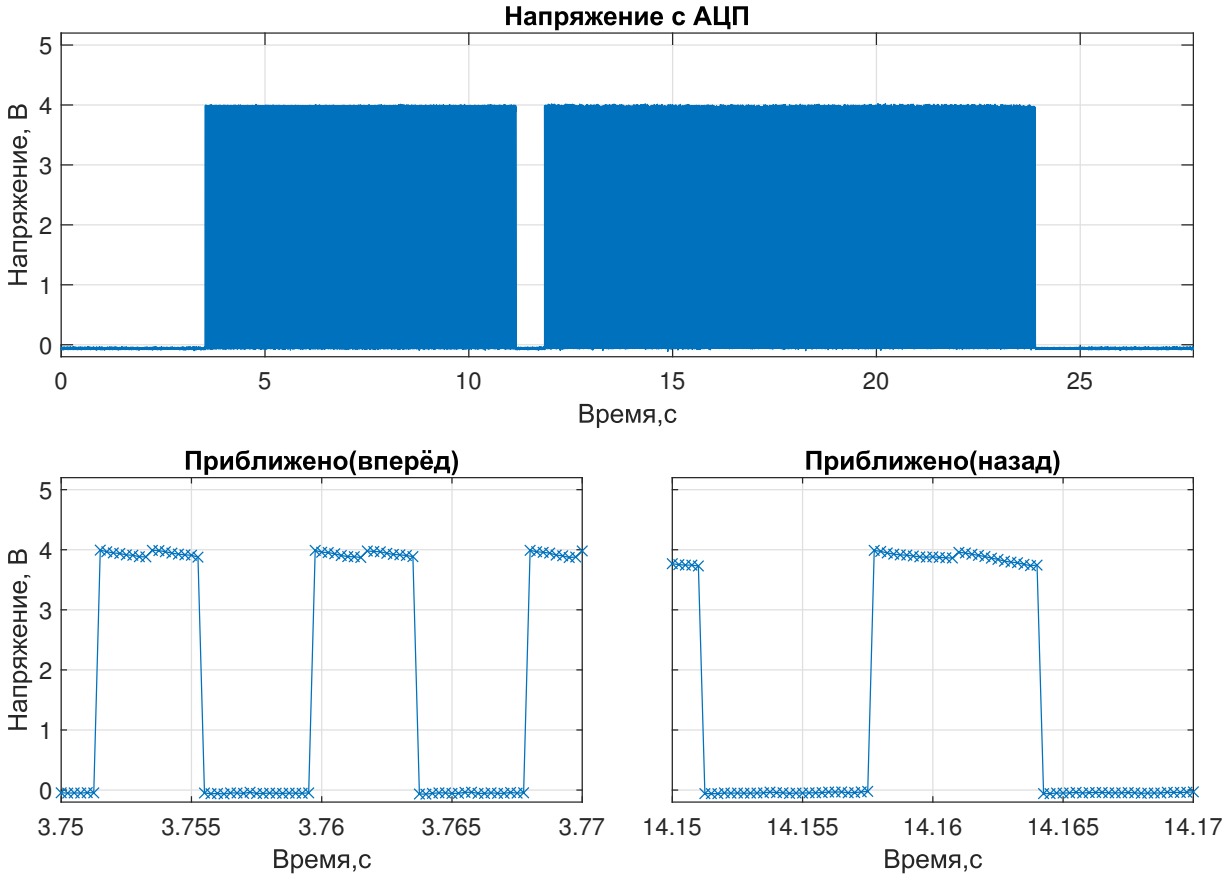


Рисунок 2.2 — Записанные значения напряжения

Далее произведена обработка - сделано насыщение - когда напряжение выше определённого значения - вывести 1, когда ниже - вывести 0 (записывается в exp.o)

После этого определён момент первого сигнала (первая единица в exp.o), момент последнего сигнала и примерно середина паузы, когда каретка прошла вперёд, но не двинулась назад.

```
for i = 1:length(exp.time)
if exp.V(i)>0.35
exp.o(i)=1;
% elseif k(i)<0.24
else
exp.o(i)=0;
end
end
```

```
exp.startidx=find(exp.o==1,1); % Первое ненулевое значение
exp.starttime=exp.time(exp.startidx);
exp.forwardidx=find(exp.time>exp.time(exp.startidx)+t1+(t2-t1)/2,1);
exp.forwardtime=exp.time(exp.forwardidx);
exp.endidx=find(exp.o==1,1,'last');
exp.endtime=exp.time(exp.endidx);

disp([exp.startidx exp.starttime; ...
exp.forwardidx exp.forwardtime; ...
exp.endidx exp.endtime])
% Вывести момент пуска,
% примерно середину паузы,
% время последнего сигнала (индекс-время)
```

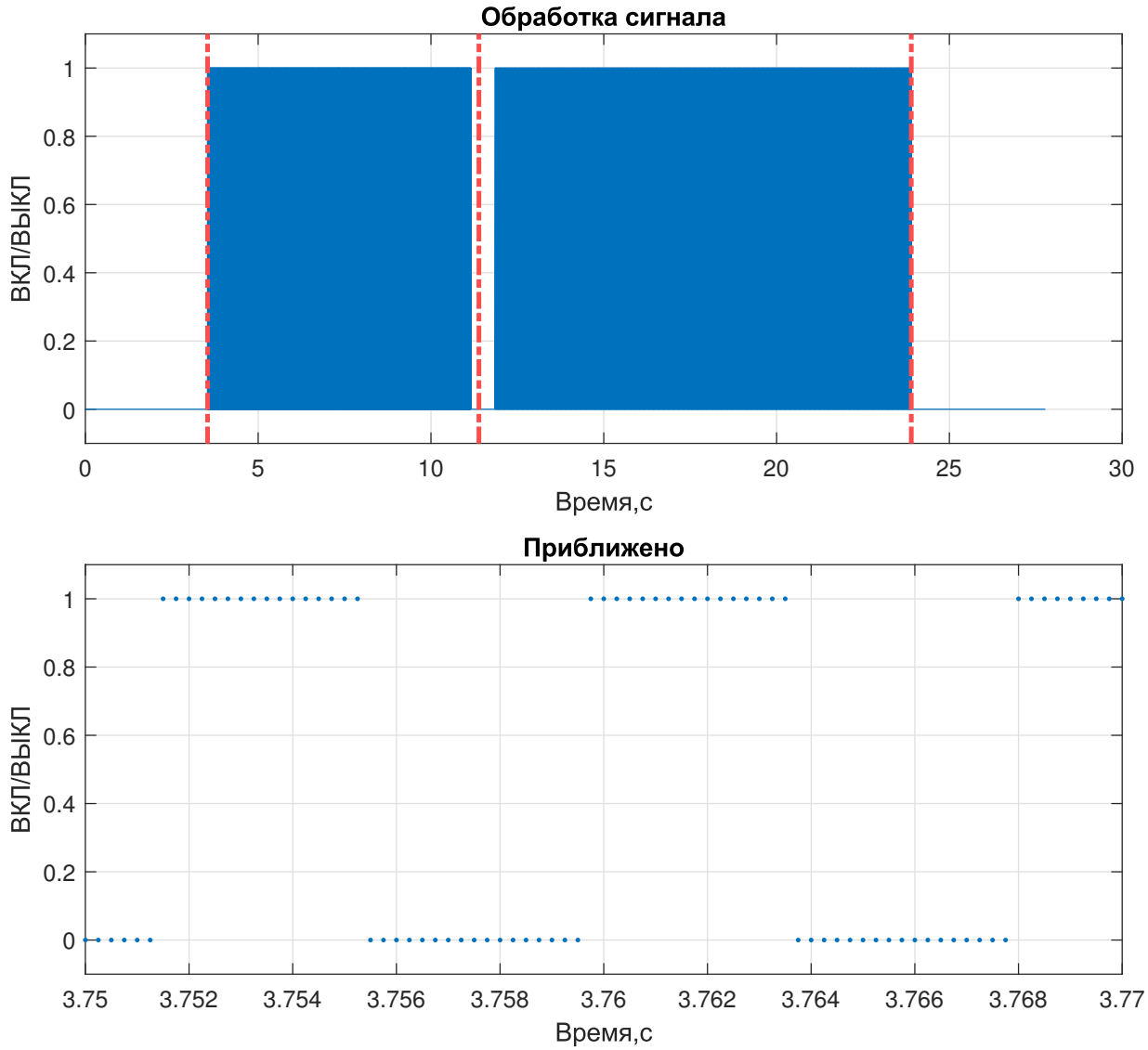
```
14147      3.5365
45548      11.387
95593      23.898
```

```
figure;
tiledlayout("flow",'TileSpacing','compact','Padding','compact')

nexttile
plot(exp.time,exp.o,'-')
xline([exp.starttime exp.forwardtime exp.endtime],'Color','red','LineStyle','-.','
gostplot("Обработка сигнала","Время,с","ВКЛ/ВЫКЛ",keep_color=true)
ylim([-0.1 1.1])

nexttile
plot(exp.time,exp.o,'.')
hold on
gostplot("Приближено","Время,с","ВКЛ/ВЫКЛ",keep_color=true)
grid on
xlim([t1/2 t1/2+0.02])
```

```
ylim([-0.1 1.1])
```



Число шагов, сделанных при движении вперёд и назад считаем так - сколько раз было переходов 0-1 (rise trigger), столько было сигналов на обмотку. Одно изменение сигнала на обмотку соответствует двум шагам (рис. 1.2)

```
% Вперёд
k=0;
for i = 1:exp.forwardidx
if exp.o(i)<exp.o(i+1)
k=k+1;
end
```

```
end
exp.nStepsForward = k*2;
% одно включение на два шага - шагов в два раза больше

k=0;
for i = exp.forwardidx:exp.endidx
if exp.o(i)<exp.o(i+1)
k=k+1;
end
end
exp.nStepsBack = k*2;
disp([exp.nStepsForward exp.nStepsBack;]) % Число импульсов вперёд и назад - выв
```

```
        1844        1844
```

Получили, что число шагов соответствует заданному при движении и вперёд, и назад.

Время одного шага определим, посчитав, сколько времени длится подаваемый сигнал (в течении движения вперёд и назад отдельно), и разделим на число сигналов

```
%Фактическое, расчётное t1 и погрешность в %
exp.t1=exp.time(find(exp.o(1:exp.forwardidx)==1,1,'last'))
-exp.time(find(exp.o(1:exp.forwardidx)==1,1));

%Фактическое, расчётное stepDelay_mks и погрешность в %
exp.stepDelay_mks(1)=exp.t1/exp.nStepsForward*1e6/2;
% Здесь снова делим на 2 - задержка вычислялась для 1/2 шага

% Аналогично для заднего хода
exp.t2=exp.time(exp.forwardidx+find(exp.o(exp.forwardidx:exp.endidx)==1,1,'las
-exp.time(exp.forwardidx+find(exp.o(exp.forwardidx:exp.endidx)==1,1));
exp.stepDelay_mks(2)=exp.t2/exp.nStepsForward*1e6/2;
```

Фактическое, расчётное t1 и погрешность в %

```
disp([exp.t1 t1 100*(1-exp.t1/t1)])
```

```
    7.6177         7.5        -1.57
```

Фактическое, расчётное stepDelay_mks вперёд и погрешность в %

```
disp([exp.stepDelay_mks(1) stepDelay_mks(1) 100*(1-(exp.stepDelay_mks(1))/ste
```

```
    2065.6         2033      -1.6011
```

Аналогично для заднего хода: время движения

```
disp([exp.t2 t2 100*(1-exp.t2/t2)])
```

```
    12.04          8.2       -46.826
```

(Большая погрешность !! установка была поломана)

Время шага

```
disp([exp.stepDelay_mks(2) stepDelay_mks(2) 100*(1-(exp.stepDelay_mks(2))/ste
```

```
    3264.6         3226      -1.1957
```

(Большая погрешность по времени движения, но сигналы на моторчик посылаются правильно - тут погрешность <2%)

3 Вывод

В ходе выполнения лабораторной работы были изучены устройство и принципы работы шагового двигателя, основы программного управления для систем автоматического управления. Разработана управляющая программа для микроконтроллера Arduino, обеспечивающая перемещение каретки на заданное расстояние с заданной скоростью.

В результате эксперимента, в ходе которого было измерено напряжение на одной из обмоток шагового двигателя, были получены следующие результаты:

Таблица 3.1 — Погрешности результатов

Величина	Эксперимент	Расчёт	Погрешность, %
Число шагов вперёд	1844	1844	0
Число шагов назад	1844	1844	0
Время движения вперёд, с	7.62	7.5	1.57
Время движения назад, с	12.04	8.2	46.83
Время сигнала (вперёд), мкс	2065.6	2033	1.6
Время сигнала (назад), мкс	324.6	3226	1.1957

Сравнение с расчётными данными показывает, что при движении назад время движение не соответствует заданному программой, что связано с неисправностью установки. По малым погрешностям по времени сигнала можно судить, что управлвющая программа составлена корректно.