2. ВТОРОЙ ЭТАП

Задачи второго этапа

Во втором этапе вам будут предложены задания, которые не только проверят вашу готовность к финалу, но и помогут к нему подготовиться. Задание финала будет связано с моделированием в САПР некоего устройства, изготовлением его деталей на станках с ЧПУ (3D-принтерах, лазерном и фрезерном), сборкой и наладкой электронной начинки и, наконец, программированием полученного устройства. Само проектируемое устройство будет некой разновидностью координатного устройства с ЧПУ. В задании понадобится также программировать на ПК (Python или C++), используя библиотеку машинного зрения OpenCV.

Команда должна состоять не более чем из 3-х человек, их роли условно обозначены как "конструктор, технолог, электронщик, программист". Желательно. чтобы каждый участник до какой-то степени сочетал несколько ролей, например, в начале работы лучше иметь больше конструкторов, ближе к концу - больше электронщиков и программистов. Совокупность знаний и умений участников команды должна включать:

Конструктор/технолог:

Умение работать в САПР (любые из: Autodesk Inventor, Fusion 360, PTC Creo, Компас 3D, другие - по согласованию с организаторами), опыт конструирования технических устройств. Понимание особенностей моделирования под конкретные технологии прототипирования.

Практический опыт изготовления изделий на 3D-принтере, лазерном станке, фрезерном станке с ЧПУ, пост-обработка с использованием ручного и станочного инструмента, сборка. НЕ требуется обязательно уметь использовать все эти технологии, но разнообразие доступных технологий очевидно облегчает задачу и позволит быстрее получить качественный результат.

Электронщик/программист:

Знание электроники (Ардуино, шаговые двигатели, сервоприводы, работа с распространенными типами датчиков), пайка и монтаж. Программирование микроконтроллеров (например, Ардуино). Программирование на ПК, на любом высокоуровневом языке (предпочтительно - Python и/или C++)

Общие навыки:

Навыки командной работы (совместное планирование, распределение работ). Структура задания.

Задание 2-го тура состоит из задач в Stepik и практической части. В задачах на Степике имеются разделы для конструктора (проектирование в САПР) и для электронщика- программиста (задачи по основам электроники и по программированию Ардуино). Практическая часть предполагает конструирование и изготовление

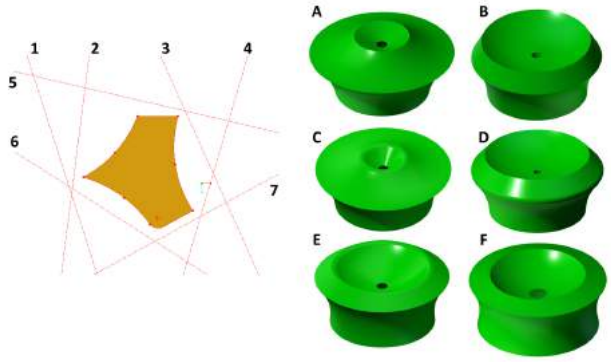
узла или упрощенного прототипа устройства, сходного с конструкцией из финального тура, также с подзадачами для конструктора и для электронщика/программиста.

Курс на Stepik "ОНТИ 18/19. Передовые производственные технологии. Этап 2."кроме (отыгранных) заданий, содержит полезные ссылки, тренировочные задания и материалы для хакатонов.

3.1. Работа в САПР: общее понимание и базовое моделирование

Задача 3.1.1. (2 балла)

Каждое из показанных справа тел было получено из эскиза слева, вращением плоской фигуры вокруг одной из осей:

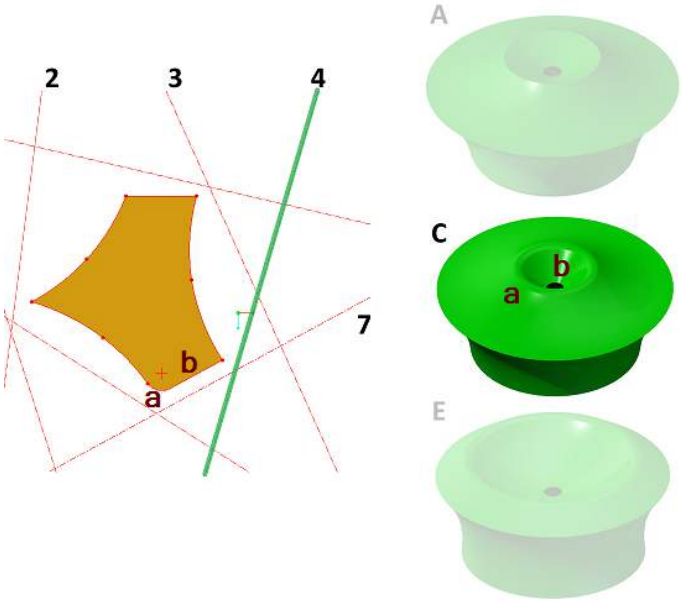


1. Тело A	а. Ось 5
2. Тело B	б. Ось 3
3. Тело C	в. Ось 6
4. Тело D	г. Ось 7
5. Тело E	д. Ось 2
6. Тело F	е. Ось 4
7. Не используется	ж. Ось 1

Решение

По эскизу определяем характерные элементы, соответствие с которыми нужно искать в телах вращения. Для каждого из показанных тел определяем ось, если надо - проверяем несколько осей, исключая неподходящие. Например, объект C имеет следующие характерные признаки:

1. сглаженный край верхнего "кратера", которому должен соответствовать скругленный угол в нижней части контура
2. короткий прямой склон "кратера", угол наклона и малый диаметр отверстия говорят о том, что "кратер" образован отрезком b, а ось должна проходить рядом с правым концом этого отрезка, под углом около 45 градусов к нему.



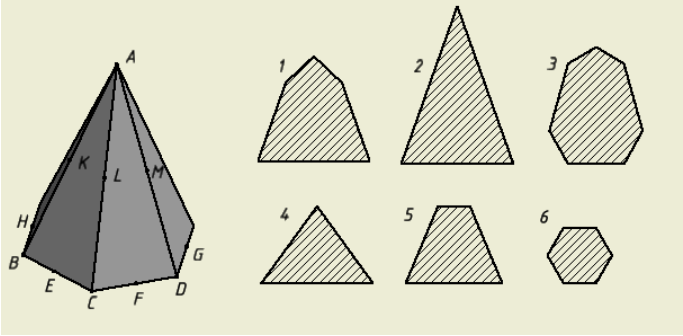
Эти признаки однозначно определяют **ось 4** как нужную нам ось. Аналогично определяются оси для каждого из остальных тел.

Ответ:

1. Тело A	б. Ось 3
2. Тело B	г. Ось 7
3. Тело C	е. Ось 4
4. Тело D	а. Ось 5
5. Тело E	д. Ось 2
6. Тело F	ж. Ось 1
7. Не используется	в. Ось 6

Задача 3.1.2. Геометрия сечений (2 балла)

Имеется пирамида, некоторые вершины и середины ребер которой обозначены буквами. Построено несколько плоскостей сечения, каждая из которых проходит, по крайней мере, через 3 точки и обозначается по этим точкам (например, "плоскость BLD"). Эти плоскости создали следующие сечения. Для каждого сечения укажите соответствующую секущую плоскость (еще 3 сечения этой же пирамиды Вы увидите на следующем шаге):

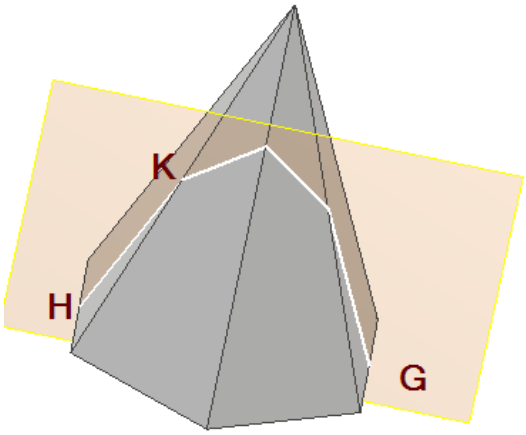


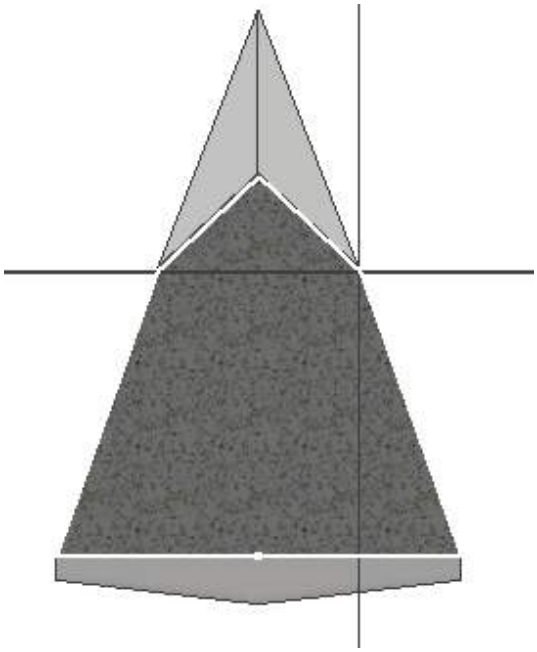
1. Плоскость FGK — а. Сечение 1
2. Плоскость GHK — б. Сечение 3
3. Плоскость ABD — в. Сечение 2

Решение

Обратите внимание, что каждое из сечений данного тела может быть получено при разрезании тела по нескольким разным плоскостям, а сами плоскости могут быть обозначены разными комбинациями букв. Поэтому начинаем не с рассматривания сечений, а с построения плоскостей, предлагаемых как варианты ответов.

Например, первая из предложенных плоскостей, GHK, пройдет, как показано на рисунке. Сразу становится очевидно, что сечением в этом случае становится фигура 1:



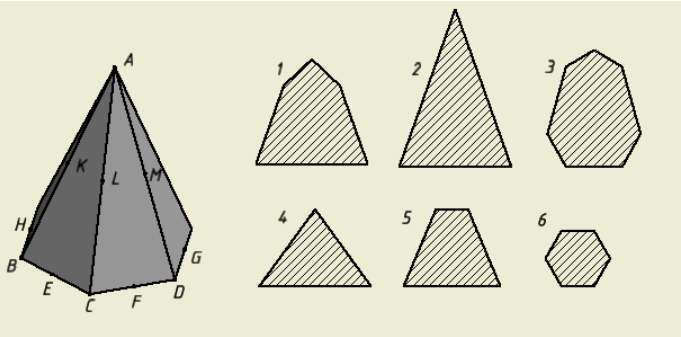


Ответ:

1. Плоскость FGK — б. Сечение 3
2. Плоскость GHK — а. Сечение 1
3. Плоскость ABD — в. Сечение 2

Задача 3.1.3. Геометрия сечений (1 балл)

Продолжаем предыдущую задачу. Сопоставьте еще 3 плоскости и 3 сечения пирамиды:



1. Плоскость EGM
2. Плоскость BDL
3. Плоскость KLM

а. Сечение 6
б. Сечение 4
в. Сечение 5

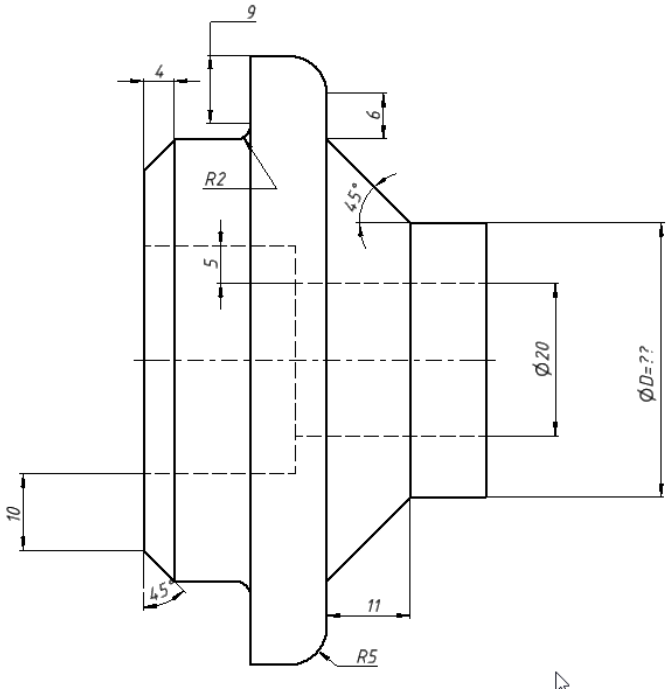
Ответ:

1. Плоскость EGM
2. Плоскость BDL
3. Плоскость KLM

в. Сечение 5
б. Сечение 4
а. Сечение 6

Задача 3.1.4. (1 балл)

По размерам, имеющимся на чертеже (все размеры даны в мм), определите диаметр d:

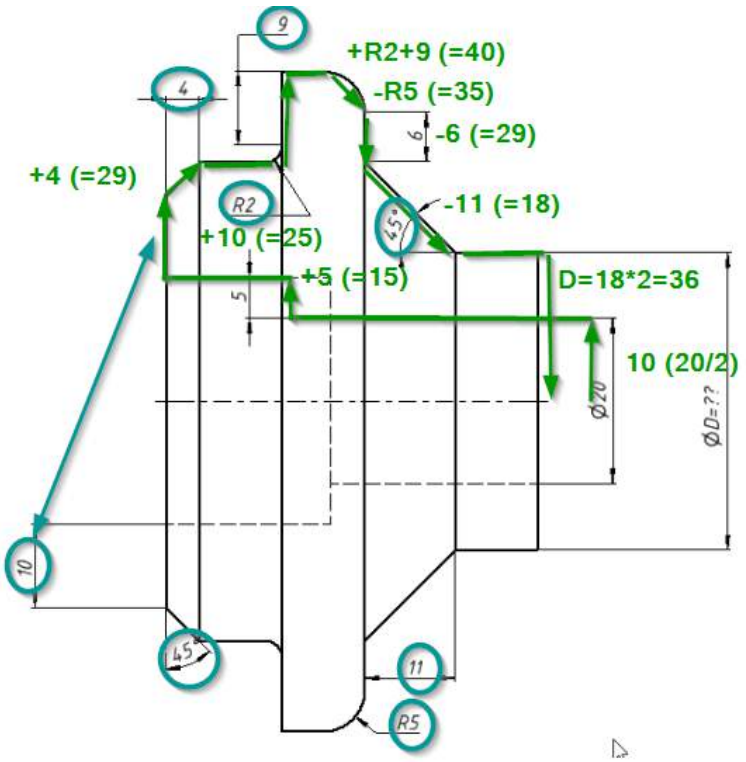


Вычислите d, введите только число (целое).

Решение

В задачах этого типа вас посылают в путешествие вокруг детали, начиная с какого-то известного размера, карабкаясь по цепочке размеров, добавляя или вычитая очередной размер. Препятствия, которые вам могут встретиться в пути:

- Размер поставлен не с той стороны, где вы его ожидаете увидеть,
- Вы "залезаете"или "съезжаете"по склону с наклоном 45 или 30 градусов, и вместо высоты склона, вам дана его ширина. Вычислять хитрые синусы/косинусы не понадобится, но кое-что про свойства треугольников надо знать,
- Вместо вертикальной стенки, вам могут встретиться скругления с заданным радиусом,
- Вы можете забыть в нужный момент перейти от радиуса к диаметру или обратно.

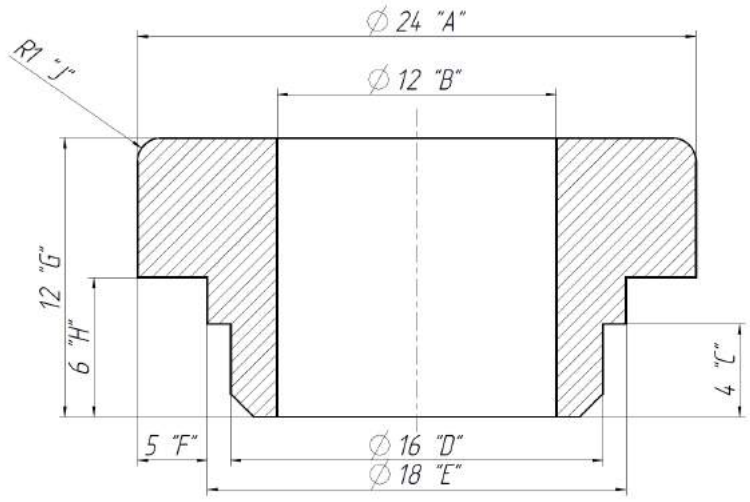


Как видно из рисунка, в данной задаче ответом будет число 36.

Ответ: 36.

Задача 3.1.5. (1 балл)

На этом чертеже неправильно указан один из размеров:



Найдите неверный размер и введите для него правильное значение, указав букву размера и верное значение в формате «буква размера>=<размер>"(например A=24). Все буквы прописные.

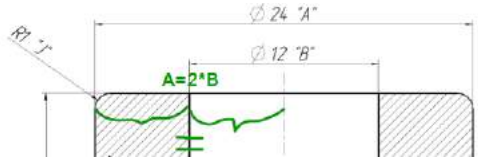
Решение

Назовем избыточным размер, который можно вычислить из других размеров, имеющихся на чертеже. Если такая цепочка размеров присутствует, то избыточным можно считать, на выбор, любой из входящих в нее размеров. При этом совсем необязательно, чтобы избыточный размер оказался неправильным. На самом деле, при генерации чертежей в САПР по 3D-модели наставить лишних размеров очень легко и просто, но нужно специально постараться (вручную изменяя размеры), чтобы один из этих размеров стал неправильным.

В данном примере есть только одна избыточная цепочка - это размеры A, E и F. Эти размеры должны удовлетворять соотношению $2 \cdot F = (A - E)$, однако на чертеже это соотношение не выполняется: $2 \cdot F = 10$, а $A - E = 24 - 18 = 6$. Значит, действительно, один из размеров в цепочке задан неверно. Очень хочется сразу решить, что неправильным (и лишним) является размер F и его настоящее значение должно быть $F = (A - E)/2 = 3$ (мм).

Однако это пока только гипотеза, с тем же успехом ошибка может быть в размере A или в размере E. Попробовать выяснить, который из этих трех размеров ошибочен можно, только сопоставляя их с другими размерами чертежа. Поэтому проверим каждый из размеров A, E, F, учитывая, что по условию задачи неправильный размер есть только один:

- Предположим, что ошибочен размер E. В этом случае его значение должно быть: $E = A - 2 \cdot F = 14$ мм, что не согласуется с чертежом: размер $D = 16$ должен быть меньше E.
- Предположим, что ошибочен размер A. Но на чертеже мы видим, что с показанными на чертеже значениями $A = 2 \cdot B$, и это подтверждается визуально равенством отмеченных отрезков. Если принять что $A = E + 2 \cdot F = 28$, то пропорции чертежа существенно изменились бы.



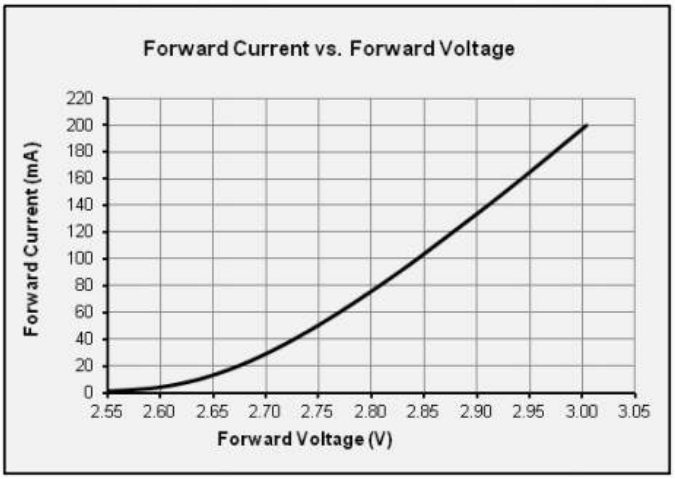
Таким образом, "неправильным"размером действительно является F и ответ, ожидаемый Stepik'ом: $F = 3$.

Ответ: F=3.

3.2. Основы электроники

Задача 3.2.1. (3 балла)

В качестве фар машинки-робота использован осветительный светодиод LM561C, запитываемый от 2-х баночного LiPo аккумулятора (7.4 В). Последовательно со светодиодом, в цепь также включен ограничительный резистор. Используя приведенную ниже вольт-амперную характеристику светодиода, выберите номинал резистора так, чтобы ток через светодиод был по возможности ближе к 100 мА, но не превышал этого значения.



a. Выберите номинал ограничительного резистора из стандартного ряда номиналов:

1. 33 Ом
2. 39 Ом
3. 47 Ом
4. 56 Ом
5. 68 Ом
6. 82 Ом
7. 100 Ом
8. 120 Ом

б. Максимальная рассеиваемая мощность этого резистора не должна быть менее чем

1. 0.125 Вт
2. 0.25 Вт
3. 0.5 Вт
4. 0.75 Вт
5. 1 Вт
6. 2 Вт

Решение

В этой задаче вам надо понимать закон Ома и знать, что такое вольт-амперная характеристика.

В отличие от, например, резисторов, светодиоды являются нелинейными электронными компонентами. Ток через светодиод почти не протекает при малых значениях напряжения, находится в рабочем диапазоне в очень узком диапазоне напряжений, а при дальнейшем увеличении напряжения резко нарастает, до перегрева и выхода светодиода из строя. График зависимости тока от напряжения называется вольт-амперной характеристикой устройства.

Как видно из графика, требуемый ток в 100 мА получается при напряжении на диоде 2.85 В. Однако же на входе у нас 7.4 В, поэтому из них $U_R = 7.4 - 2.85 = 4.55$ В должны падать на ограничительном резисторе. Поскольку резистор и светодиод соединены последовательно, ток через резистор составляет те же 100 мА. По закону Ома:

$$R = U_R / I = 4.55 \text{ В} / 0.1 \text{ А} = 45.5 \text{ Ом}$$

Однако резисторы изготавливаются определенных номиналов, и Stepik предлагает выбрать один из них (33, 39, 47, 56, 68 Ом). Выбираем резистор с ближайшим большим значением - 47 Ом.

Какова же должна быть максимальная рассеиваемая мощность этого резистора? Мощность вычисляется по формуле:

$$P = U \cdot I = 4.55 \text{ В} \cdot 0.1 \text{ А} \approx 0.45 \text{ Вт}$$

Подбираем ближайшее большее значение (0.125, 0.25, 0.5, 0.75, 1, 2 Вт): 0.5 Вт.

Ответ: а - 3, б - 3.

3.3. Программируем Arduino: простые схемы и задачи

Задача 3.3.1. Простой blink (2 балла)

В этом шаге вам нужно будет изменить и проверить работу простейшего примера blink из стандартного набора примеров Arduino IDE, для этого следует изменить номер пина и интервал мигания, как указано в шаблоне!

Такие задачи стали возможны благодаря мини-симулятору Arduino, который имитирует действие некоторых функций библиотеки Arduino, изменяя состояние хранимых в памяти "пинов"и сообщая о каждом изменении платформе Stepik, чтобы проверить правильность вашего решения. Вы пишете код КАК БЫ для Ардуино, на самом деле он выполняется в Степике, взаимодействуя с платформой через стандартный ввод-вывод, как и любая другая задача на программирования.

Вам НЕ НАДО трогать что-нибудь в шаблоне, кроме функций setup() и loop(). Разумеется, для решения более сложных задач могут понадобиться дополнительные функции или переменные, которые вы можете определить там же. Можно использовать большинство стандартных библиотек C++, но никакие библиотеки, специфические для Arduino, применить не получится. Мини-симулятор настраивается под задачу, и в нем, как правило, не будут работать никакие функции, не относящиеся конкретно к данной задачи.

Функции библиотеки Arduino, которые можно использовать в этой задаче:

void digitalWrite(int pin, int state): изменяет хранимое в памяти состояние пинов, и сообщает в Stepik о каждом изменении, вместе с отметкой времени. Если был задан неверный номер пина, либо этот пин не был переведен в режим OUTPUT, либо состояние пина не изменилось, то ничего не происходит.

void pinMode(int pin, int mode): устанавливает режим работы пина: INPUT, OUTPUT или INPUT_PULLUP.

void delay(int n): "Задержка"на n миллисекунд, а на самом деле - просто увеличение внутреннего счетчика времени. Мини-симулятор следит за временем, хотя, в отличие от реального микроконтроллера, счетчик времени увеличивается только и исключительно вызовами функции delay(). Любые операции, между которыми нет delay(), считаются выполняющимися мгновенно.

Если вы хотите испытать свой код на реальной схеме с Ардуино, просто перенесите в среду Ардуино написанный вами код и он, МОЖЕТ БЫТЬ, даже будет работать. Разумеется, вам сначала придется собрать соответствующую описанию схему.

Пример программы-решения

Ниже представлено решение на языке C++

```
int LED_PIN;   // каким пином мигаем
int ON_PERIOD; // время свечения, мс
int OFF_PERIOD; // интервал между миганиями, мс
void setup()
{
    pinMode(LED_PIN,OUTPUT);
}
void loop()
{
    digitalWrite(LED_PIN, HIGH);
    delay(ON_PERIOD);
    digitalWrite(LED_PIN, LOW);
    delay(OFF_PERIOD);
}
```

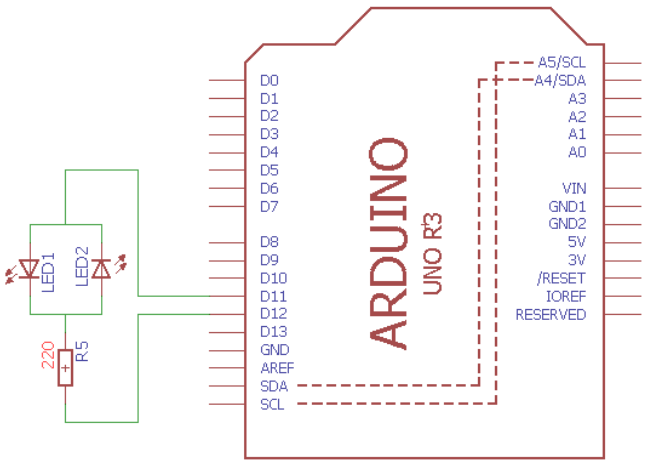
Пример программы-решения

Ниже представлено решение на языке C++

```
int LED_PIN;    // каким пином мигаем
int ON_PERIOD; // время свечения, мс
int OFF_PERIOD; // интервал между миганиями, мс
void setup()
{
    pinMode(LED_PIN,OUTPUT);
}
void loop()
{
    digitalWrite(LED_PIN, HIGH);
    delay(ON_PERIOD);
    digitalWrite(LED_PIN, LOW);
    delay(OFF_PERIOD);
}
```

Задача 3.3.2. Полицейский маячок (BLINK со встречным подключением светодиодов) (3 балла)

Два светодиода, красный и зеленый, подключены к Ардуино, как показано на схеме:



Напишите программу, которая попеременно мигает красным и зеленым светодиодами, без "темных"интервалов. Каждый светодиод включен LED_PERIOD миллисекунд (интервал меняется от теста к тесту). Какой из светодиодов зажигается первым, роли не играет. Имеются функции pinMode(), digitalWrite(), delay().

Решение

Это еще одна супер-легкая задача, рассчитанная на то, чтобы дать вам разобраться с написанием кода для мини-симулятора. Раз диоды подключены "встречно программа должна попеременно подавать LOW на D11 и HIGH на D12 (включен LED2) или HIGH на D11 и LOW на D12 (включен LED2). Для мини-симулятора "одновременными"считаются действия, выполняемые без вызова delay() между ними.

Прежде, чем писать собственно код для переключения, не забываем инициализировать соответствующие пины как выходные. При этом обязательно использовать именованные константы, приведенные в шаблоне кода, а не числа:

```
void setup()
{
    pinMode(LED_PIN1, OUTPUT);
        pinMode(LED_PIN2, OUTPUT);
}
```

Вот самое простое (и тупое) решение для поочередного мигания (и опять-таки, обязательно используем параметр LED_PERIOD, приведенный в шаблоне кода. Он будет принимать разные значения для разных тестов, и иначе программа правильный ответ не выдаст):

```
void loop()
{
    digitalWrite(LED_PIN1, HIGH);
    digitalWrite(LED_PIN2, LOW);
    delay(LED_PERIOD);
    digitalWrite(LED_PIN1, LOW;
    digitalWrite(LED_PIN2, HIGH));
    delay(LED_PERIOD);
}
```

А вот чуть более интересное решение с использованием переменной:

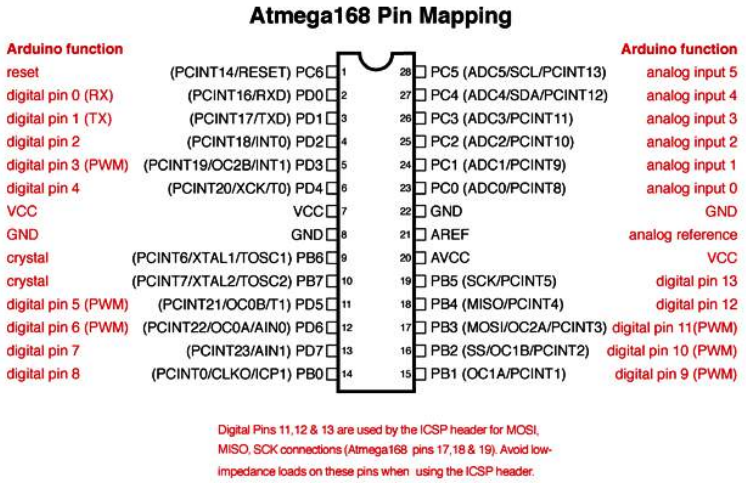
```
bool b = true;

void loop()
{
    digitalWrite(LED_PIN1, b);
    b = !b;
    digitalWrite(LED_PIN2, b);
    delay(LED_PERIOD);
}
```

Отступление (для продвинутых):

В реальном контроллере между соседними вызовами digitalWrite() будет, разумеется, некая микросекундная задержка. Для данной задачи это никакой роли не играет, но в некоторых ситуациях может быть важным. Используя только вызовы из библиотеки Ардуино, невозможно строго одновременно управлять несколькими пинами.

Сам микроконтроллер ATMEGA, тем не менее, это делать позволяет. На аппаратном уровне, "пины" (входы/выходы) контроллеры объединены в 8-битовые "порты", доступные как регистры контроллера, а на уровне языка C - как специальные переменные. Таким образом, можно строго одновременно, за 1 такт микроконтроллера, менять состояние до 8 пинов!

Если вы загуглите "ATMEGA ArduinoUNO pin mapping то найдете, например, вот такую картинку:



В контексте данной задачи, нам важен тот факт,что пинам Arduino D11 и D12 соответствуют биты 3 и 4 порта B контроллера ATMEGA. На языке C, к ним можно обратиться следующим образом (причем это действует не только в "настоящей"среде разработки AVRStudio, но и в среде Arduino!):

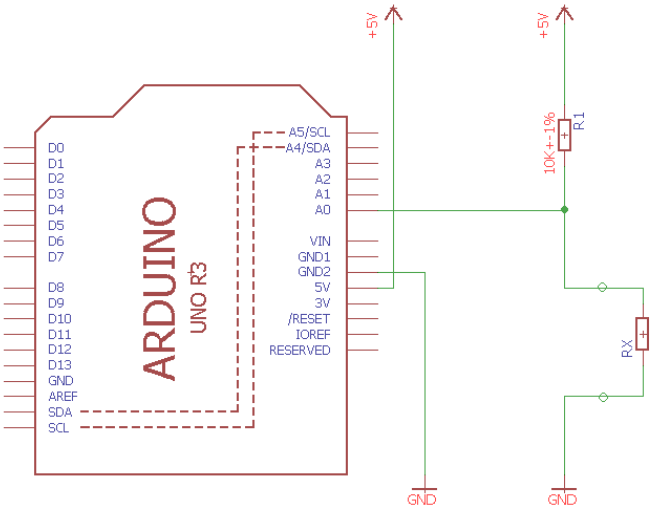
```
void setup()
{
   DDRB =  0b00011000;   // заменяет pinMode, устанавливая PB3 и PB4 в OUTPUT
   PORTA = 0b00010000;   // начальное состояние: PB3(D11) = 0, PB4(D12) = 1
}
void loop()
{
   PORTA ^= 0b00011000;  // одной командой переключаем оба пина!
   delay(LED_PERIOD);
}
```

Такая программа не только выполняется на реальном контроллере гораздо быстрее, но и занимает меньше места в памяти программ, чем код с вызовами функций библиотеки Ардуино. Обратите внимание, что если бы светодиоды были подключены, например, к пинам D7 и D8, то показанный выше трюк у нас бы не прошел, т.к. эти пины относятся к разным портам и не могут изменяться одной командой.

К сожалению, в мини-симуляторе низкоуровневая работа с портами не поддерживается, поэтому приведенный выше код не может быть решением данной задачи в Stepik. Тем не менее, любознательным среди вас составители задания советуют попробовать такую программку на реальном Ардуино и, по документации к контроллеру ATMega разобраться с псевдо-переменными PORTx, PINx и DDRx.

Задача 3.3.3. Простейший омметр (2 балла)

Вам нужно реализовать на Arduino простейший омметр (измеритель сопротивления), как показано на схеме.



Для этого к пину A0 подключен делитель напряжения, состоящий из прецизионного резистора 10 кОм "сверху"и измеряемого резистора "снизу". После того, как очередной измеряемый резистор подключен к схеме, контроллер включается, программа на Arduino производит измерение и через последовательный порт выводит в виде числа значение сопротивления, в кОм, с округлением до целого и с переводом строки на конце.

Для упрощения, мы принимаем, что измеряемое сопротивление всегда целое число килоом и отсутствуют любые ошибки измерения. Все значения измеряемых резисторов находятся в диапазоне от 10 кОм до 90 кОм (т.е. допускающем их сравнительно точное измерение при 10кОм подтягивающем резисторе).

Напишите программу для мини-симулятора Arduino, которая выполняет измерения.

Данные для измерения поступают из входного потока и расшифровываются мини-симулятором, а вычисленные вашей программой и отправленные на Serial.println() значения передаются в выходной поток для проверки. Мини-симулятор однократно вызывает функцию loop() для выполнения каждого измерения.

Разрешается пользоваться функциями pinMode(), analogRead() , а также упрощенным классом Serial с методами begin(), print() и println(). Передача данных должна происходить на скорости 9600 бод.

Решение

Очевидно, что это задача на использование имеющегося в микроконтроллере ATMEGA аналого-цифрового преобразователя (АЦП). На уровне библиотеки Arduino, чтение аналогового значения на входе АЦП выполняется функцией analogRead(). Эта функция возвращает значения в диапазоне от 0 (на входе - нулевое напряжение) до 1023 (напряжение питания). При подключении на аналоговый вход делителя напряжения, изображенного на схеме, напряжение на входе будет составлять:

$$V_x = V_{cc} \cdot R_x/(R_1 + R_x),$$

а значение, принятое с АЦП:

$$X_{ADC} = 1023 \cdot R_x/(R_1 + R_x)$$

Решая это уравнение относительно R_x , получаем:

$$R_x = R_1 \cdot X_{ADC}/(1023 - X_{ADC})$$

При реализации вычислений на C, всегда следует обращать внимание на правильность задания типов данных и на возможные ошибки округления. Наш код может выглядеть так:

```
#include <math.h>
Float R1 = 10.0;
void loop()
{
    int x = analogRead(A0);
    float R = R1 * x / (1023 - x);
    Serial.println(round(R));
}
```

Разумеется, нельзя забывать и про правильную инициализацию как пинов, так и последовательного порта (хотя при старте Arduino все пины уже и так находятся в режиме INPUT, желательно прописать pinMode() просто для большей ясности):

```
void setup()
{
    pinMode(A0, INPUT);
    Serial.begin(9600);
}
```

3.4. Работа в САПР: Сборки

Задача 3.4.1. (2 балла)

Соберите головоломку, используя приложенные файлы (в формате STEP):

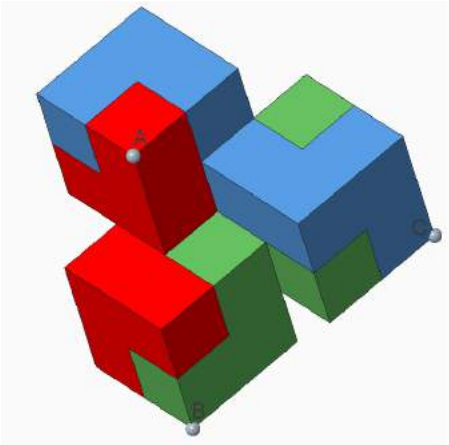
Деталь A

`https://drive.google.com/file/d/1JcXJwKiEi42S5JiICzoJdA31vLrjrEB-/view?usp=sharing`

Деталь B

`https://drive.google.com/file/d/1jTdQmNkdBCKJRBW_yd5K-0kuQvOURAFe/view?usp=sharing` Деталь C

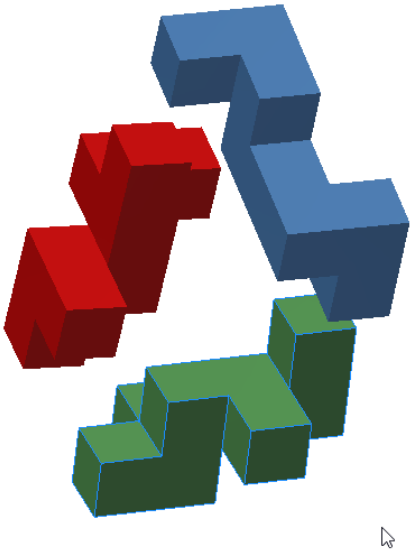
`https://drive.google.com/file/d/1prU86HaGt6WiuD1ZvGuyYfNLL1i98_tx/view?usp=sharing`



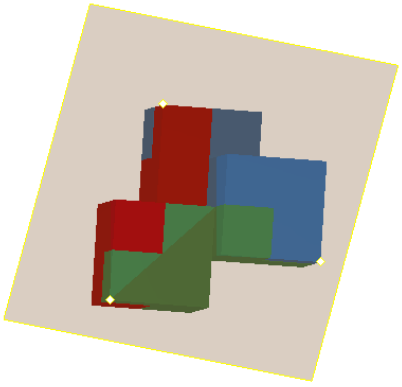
Какова площадь (в мм^2) треугольника, построенного по точкам ABC, указанным на рисунке?

Решение

Создаем сборочную модель, загружаем в нее 3 детали в формате STEP, скачанные по ссылкам. Крутим каждую деталь (в Autodesk Inventor - команда "свободный поворот"), пока она не займет положение, узнаваемое по верхнему рисунку.

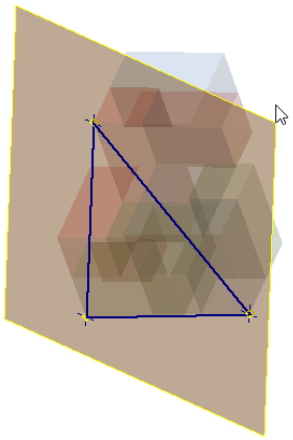


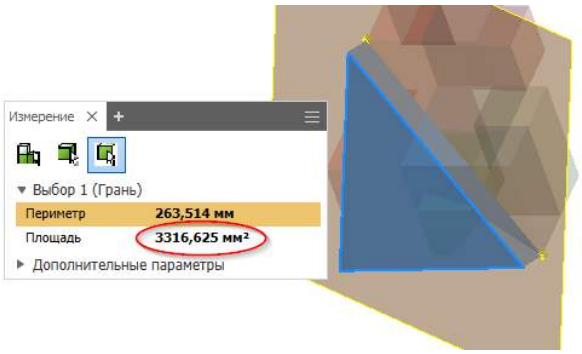
Соединяем детали сборочными зависимостями по граням, ребрам или точкам. Обычно требуется 3 зависимости для соединения каждых двух деталей.



Когда головоломка собрана, строим плоскость по 3-м точкам, показанным в задании. Учитывая, что работа идет в сборке, дальнейшее будет различаться в разных САПР.

Так, в Autodesk Inventor придется создать новую деталь в контексте сборки. После этого В плоскости строим эскиз. а в эскизе - треугольник, построенный по проекциям этих 3х точек. Слегка выдавливаем контур, чтобы измерить его площадь.

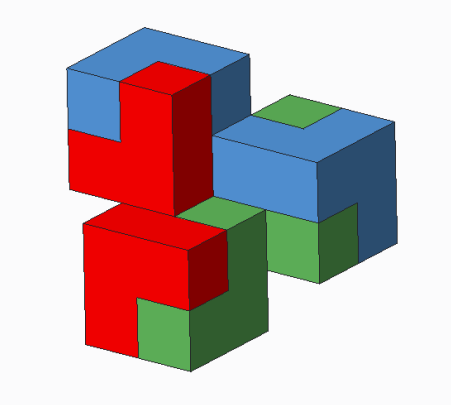




Ответ: 3316.625 ± 1

Задача 3.4.2. (2 балла)

В головоломке, собранной в предыдущей задаче, какая площадь поверхности фигуры С (красная деталь), соприкасается с другими деталями?



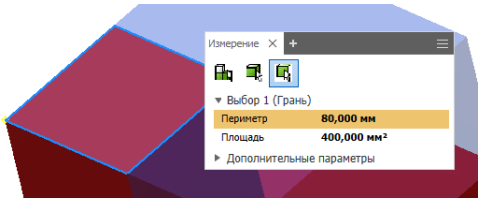
Введите площадь, в мм^2 (только число).

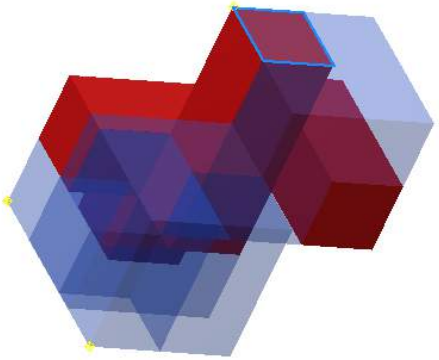
Решение

Для начала, посмотрим площадь единичного квадрата на торце элемента головоломки. Как видно, она составляет 400 мм^2.

Для облегчения подсчетов, делаем остальные 2 детали полупрозрачными. Считаем грани.

Получается 12 единичных квадратов, или 4800 мм^2.





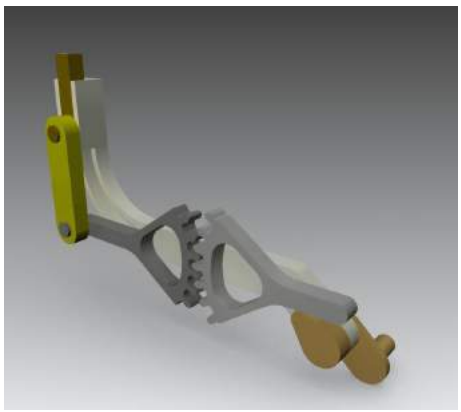
Ответ: 4800.

Задача 3.4.3. Кулачковый механизм (5 баллов)

Скачайте архив с деталями с формате STEP.

(`https://drive.google.com/file/d/1NaBgymX6Suge4ll5Ia8cB15TWLGF4S6A/view?usp=sharing`)

Соберите из этих деталей кулачковый механизм, как показано на рисунке:



Введите амплитуду движения поршня при вращении рукоятки (т.е. расстояние между крайними положениями поршня), в мм, только число, с точностью не хуже 0.1.

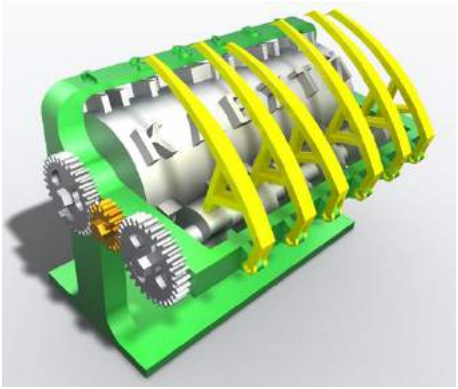
Пояснения к ответу

В задачах этого типа вам предлагается собрать механизм и вычислить диапазон движений, либо найти секретный код, определяемый по движению механизма. Требуется умение применять в сборках динамические зависимости - зубчатые передачи, кулачковые механизмы. Возможны задачи, в которых нужно изменить или добавить детали, чтобы механизм работал. В этом случае спрашивается какой-либо ключевой параметр измененной/добавленной детали. Полный процесс сборки кулачкового механизма (Вариант А) в САПР Autodesk Inventor показан в видеоуроке по адресу: `https://bit.ly/2Jih7gs`.

Ответ: 10.14 ± 0.1

Задача 3.4.4. Кодовая машина (8 баллов)

Скачайте комплект деталей (`https://drive.google.com/file/d/162uWxhMfkUMoLGutxY6GeZiJncI3fekF/view?usp=sharing`) и соберите из них показанное ниже устройство. Обратите внимание, что на зубчатых колёсах есть метки для правильной ориентации кулачкового вала относительно барабана с буквами. Если всё собрано правильно, то при вращении барабана рычаги опускаются напротив букв, составляющих слово, которое нужно ввести в ответ задания. Первый кулачок определяет первую букву слова, второй - вторую букву и так далее, всего шесть букв в слове.



Введите кодовое слово. Ответ вписываем заглавными русскими буквами. Вместо пробела на валу используется символ "_ если в коде попался этот символ, так и вводим "_ ".

Ответ: К_ЦЕЛИ.

3.5. Arduino: АЦП и делители напряжения

Задача 3.5.1. Асинхронный маячок (3 балла)

К цифровым пинам контроллера Ардуино подключены 3 светодиода разных цветов (красный, желтый и зеленый). Напишите программу, которая будет одновременно мигать этими светодиодами, причем каждым - со своим периодом и скважностью. В начальный момент времени все 3 светодиода должны включиться одновременно.

Входные данные мини-симулятор считывает автоматически, но вам нужно использовать переменные, как указано в шаблоне кода. Как и в предыдущем примере, вам доступны функции pinMode(), digitalWrite() и delay().

Решение

Задачка с тремя светодиодами требует совсем другого подхода: мы должны использовать в конце функции loop() ровно одну короткую задержку (например, на 1 мс) и для каждого светодиода использовать отдельную переменную-счетчик, которая увеличивается при каждом проходе цикла. При заданных пороговых значениях счетчика светодиод может включаться или отключаться. По сравнению с фиксированной задержкой в конце цикла, время выполнения остальной части кода (проверки счетчиков и управление пинами) пренебрежимо мало.

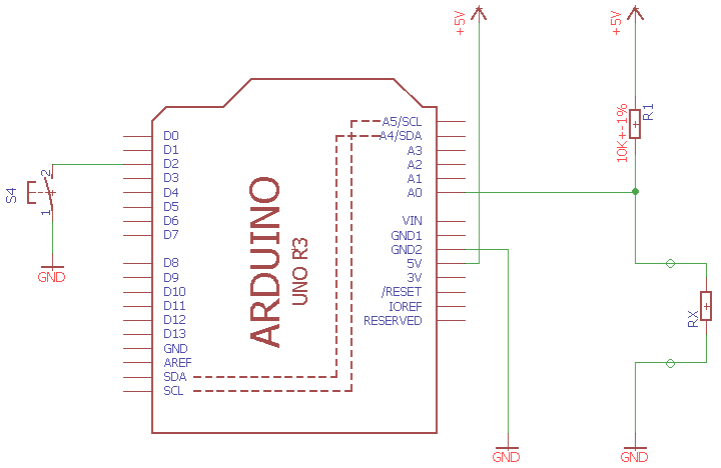
Пример программы-решения

```
int PIN_RED = 3;         // на этих пинах ваши светодиоды
int PIN_YELLOW = 4;
int PIN_GREEN = 5;

// Для каждого LED задается полный период мигания и длительность
// во включенном состояния.
// Данные считываются мини-симулятором из входного потока в эти переменные.
```

```
8       //и будут разными для каждого теста.
9
10      int RED_PERIOD, RED_ON_PERIOD;       // период мигания и длительность включенного
11      // состояния для КРАСНОГО
12      int YELLOW_PERIOD, YELLOW_ON_PERIOD; // то же для ЖЕЛТОГО
13      int GREEN_PERIOD, GREEN_ON_PERIOD;   // то же для ЗЕЛЕНОГО
14
15      int cnt_r=0, cnt_y=0, cnt_g=0; // счетчики для светодиодов
16
17      void setup()
18      {
19          pinMode(PIN_RED, OUTPUT);
20          pinMode(PIN_YELLOW, OUTPUT);
21          pinMode(PIN_GREEN, OUTPUT);
22      }
23
24      void blink_led(int pin, int &cnt, int on_period, int period)
25      {
26          if( !cnt ) //в начале периода ставим HIGH
27              digitalWrite(pin, HIGH);
28          else if( cnt >= on_period ) //по истечении времени включения - выключаем
29              digitalWrite(pin, LOW);
30          cnt = (cnt+1) % period;  // циклически увеличиваем счетчик по модулю period
31      }
32
33      void loop()
34      {
35          blink_led(PIN_RED, cnt_r, RED_ON_PERIOD, RED_PERIOD);
36          blink_led(PIN_YELLOW, cnt_y, YELLOW_ON_PERIOD, YELLOW_PERIOD);
37          blink_led(PIN_GREEN, cnt_g, GREEN_ON_PERIOD, GREEN_PERIOD);
38          delay(1);
39      }
```

Задача 3.5.2. Усовершенствованный омметр (4 балла)

В задании на прошлой неделе вам предлагалось запрограммировать простой измеритель сопротивления. Вернемся вновь к этой теме, но внеся небольшие усовершенствования:



1. Теперь между пином D2 и землей подключен нормально-разомкнутый кнопочный выключатель. Измеряемые резисторы можно заменять без выключения устройства. После того, как очередной измеряемый резистор подключен

к схеме, пользователь кратковременно нажимает кнопку. В момент отпускания кнопки, программа на Arduino производит измерение и выводит в виде числа значение сопротивления через последовательный порт. Если измерение проведено до отпускания кнопки, или более чем через 10ms после этого момента, analogRead() возвращает значение 1023 (как если бы измеряемый резистор еще не подключили).

2. Измеряемые сопротивления, в диапазоне от 10 кОм до 82 кОм, выбираются из номинального ряда E12 (Гугл в помощь!). Более того, как у настоящих резисторов, значения будут отклоняться от номиналов на случайную величину, не превышающую 10% номинала. Вашей программе нужно привести измеренное сопротивление к номиналу и вывести его в виде целого числа (в кОм) вызовом Serial.println(). Например, если ваша программа получила значение 54 кОм, то должно быть возвращено число 56 - ближайшее значение из номинального ряда. Данные должны передаваться на скорости 9600 бод.

Напишите программу для мини-симулятора Arduino, которая реализует описанный выше функционал: ждет нажатия и отпускания кнопки, проводит измерение, обрабатывает и выводит результат. Разрешается пользоваться функциями pinMode(), digitalRead(), analogRead() и delay(), а также упрощенным классом Serial с методами begin(), print() и println().

ВАЖНО: в отличие от реального Arduino, в программе для мини-симулятора любые циклы ожидания должны включать вызов функции delay(), иначе в симуляторе "время не идет"и ваша программа "зациклится не сможет дождаться никаких внешних событий.

Стандартный вывод
19 818 869 518 906 523 741 510 692 556 869 517 789 509 739 844 849 614 648 500

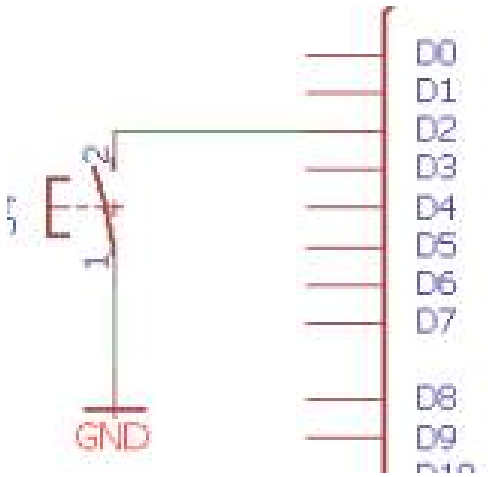
Стандартный вывод
39 56 10 82 10 27 10 22 12 56 10 33 10 27 47 47 15 18 10

Решение

Очевидно, что собственно измерение сопротивления выполняется здесь точно так же, как и в предыдущей задаче. Отличаться будет цикл ожидания нажатия кнопки до измерения и обработка ("нормализация") полученного значение после. Используя метод проектирования сверху-вниз, запишем сначала новую функцию loop(), а затем уже будем разбираться с дополнительными функциями:

```
float R1 = 10.0;
void loop()
{
    waitForButton(BUTTON_PIN);
    int x = analogRead(A0);
    float R = R1 * x / (1023 - x);
    Serial.println(normalizeResistor(R));
}
```

Чтобы поймать момент отпускания кнопки, нам сначала надо дождаться ее нажатия. Надо понимать, что при подключении кнопки между пином и землей, без внешнего подтягивающего резистора, режим пина должен быть обязательно выставлен в INPUT_PULLUP, иначе состояния пина при разомкнутой кнопке будет неопределенным. При этом, при замыкании кнопки на пине окажется 0 (LOW), а в разомкнутом состоянии - 1 (HIGH).



С учетом этого:

```
void waitForButton(int pin)
{
        while(digitalRead(pin))   // ждем нажатия кнопки
         delay(1);
        while(!digitalRead(pin))  // а затем - ее отпускания
         delay(1);
}
```

ВАЖНО: в отличие от реального Arduino, в программе для мини-симулятора любые циклы ожидания должны включать вызов функции delay(), иначе в симуляторе "время не идет"и ваша программа не сможет дождаться никаких внешних событий.

Следующее, с чем надо разобраться - это приведение измеренного значения к ближайшему стандартному значению из номинального ряда E12 (см. Википедию "Ряды номиналов радиодеталей"). Для начала, просто зададим этот ряд (в диапазоне 10 кОм - 90 кОм) в виде массива констант:

```
const int E12[] = {10, 12, 15, 18, 22, 27, 33, 39, 47, 56, 68, 82 };
```

Самый простой способ нормализовать измеренное значение - просто сравнивать его поочередно с каждым из измеренных значений. Если отклонение не превышает $\pm 10\%$, то мы нашли нужный номинал:

```
int normalizeResistor(float R)
{
    for( int i = 0; i < 12; i++ ){
        int rn = E12[i];
        if( R >= 0.9 * rn && R < 1.1 * rn ) return rn; // номинал найден
    }
    return round(R);   // не должно случиться - возвращаем без нормализации
}
```

Такой алгоритм, однако, может давать неоднозначные ответы, поскольку 10% диапазоны для соседних значений слегка перекрываются. Например: $10 \cdot 1.1 = 11$, а $12 \cdot 0.9 = 10.8$. Таким образом, измеренное значение 10.9 кОм может соответствовать, строго говоря, номиналам как 10 кОм, так и 12 кОм. В данной задаче это не приведет

к проблемам, так как генерируемые тестовые данные не будут настолько сильно отклоняться от своих номиналов, чтобы попасть в "серые зоны".

Тем не менее, чтобы избежать неоднозначности в определении номинала, можно сравнивать измеренное значение со средними арифметическими соседних номиналов. Например, для пары номиналов 12 кОм и 15 кОм естественной границей будет $(12+15)/2 = 13.5$ кОм. Вот как это можно сделать:

```
int normalizeResistor(float R)
{
  if( R < 0.9 * E12[0] ) return R; // меньше меньшего - возвращаем как есть
  if( R > 1.1 * E12[11] ) return R; // больше большего - возвращаем как есть

  for( int i = 0; i < 11; i++ ){     // не включая последний элемент
      float rn = (E12[i] + E12[i+1]) / 2.0;  // Граница между номиналами
      if( R < rn ) return E12[i]; // номинал найден
   }
   return E12[11];    // последний номинал
}
```

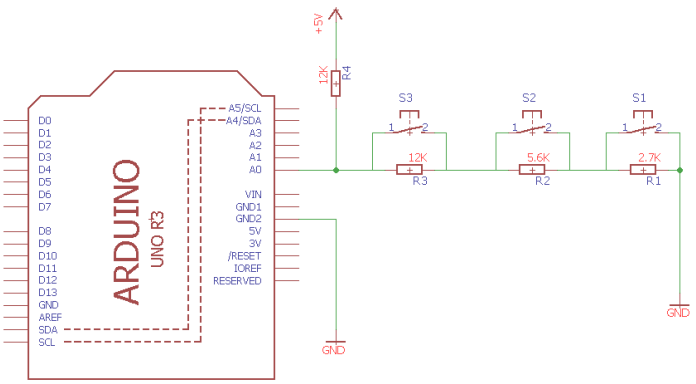
Ну и, наконец, не забываем об инициализации в функции setup(). По сравнению с предыдущей задачей, здесь добавляется установка режима INPUT_PULLUP для пина, к которому подключена кнопка:

```
void setup()
{
    pinMode(A0, INPUT);
    pinMode(BUTTON_PIN, INPUT_PULLUP);
    Serial.begin(9600);
}
```

Задача 3.5.3. Аналоговые кнопки (4 балла)

Если нужно подключить к устройству несколько кнопок, а пины контроллера приходится экономить, то подключают цепь из нескольких кнопок и резисторов к одному аналоговому входу, так что при разных комбинациях нажатий получаются разные сопротивления цепи, и, соответственно, разные напряжения на аналоговом входе.

И вот, имеется такая схема:



Напишите программу, которая сразу после включения определяет нажатые кнопки и передает через последовательный порт одно десятичное число, соответствующее их комбинации. Каждой из кнопок S1..S3 соответствует один бит в полученном числе. Например, нажатые кнопки S1 и S2 должны давать число 3 (2+1). Ожидаемая скорость передачи - 115200 бод. Программа должна учитывать, что сопротивления резисторов могут несколько (до 2.5% в каждую сторону) отклоняться от указанных на схеме значений.

В этом примере, мини-симулятор будет выполнять функцию loop() однократно для каждой комбинации нажатых кнопок. Никаких задержек и ожиданий в коде писать не надо. Используются функции analogRead(), pinMode() и упрощенный Serial, как в предыдущих заданиях.

Решение

Если нужно подключить к устройству несколько кнопок, а пины контроллера приходится экономить, то часто подключают цепь из нескольких кнопок и резисторов к одному аналоговому входу, так что при разных комбинациях нажатий получаются разные сопротивления цепи, и, соответственно, разные напряжения на аналоговом входе.

В данной схеме "верхний"резистор делителя $R4 = 12$ кОм, а "нижняя"ветка делителя состоит из 3-х последовательно соединенных резисторов ($R1 = 2.7$ К, $R2 = 5.6$ К, $R3 = 12$ К), каждый из которых может быть шунтирован при нажатии соответствующей кнопки. Для любой заданной комбинации кнопок (из 8 возможных) можно заранее вычислить напряжение на пине A0. Например, при замкнутых $S1$ и $S3$ в нижней части делителя останется только $R2 = 5.6$ К, и результат analogRead() будет равен:

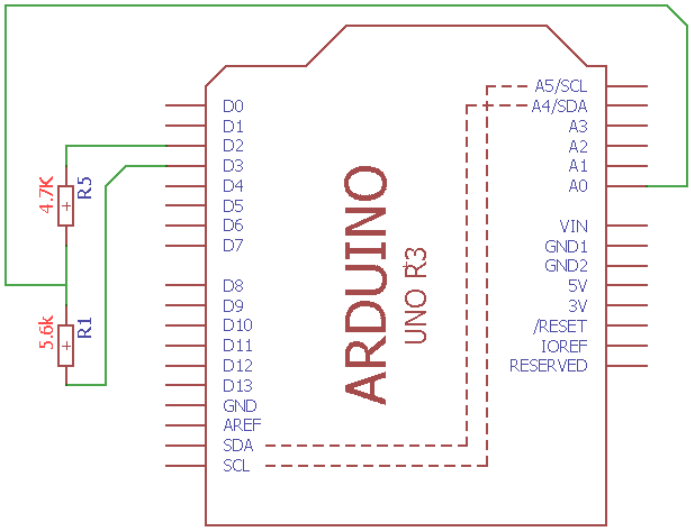
$$A = R2/(R2 + R4) \cdot 1023 = 5.6/(5.6 + 12) \cdot 1023 \approx 325$$

С учетом 2.5% допусков, диапазон значений analogRead() для этой комбинации кнопок составляет: $A_{min} = A \cdot 0.975 \approx 317$, $A_{max} = A \cdot 1.025 = 333$

Программа сравнивает полученное значение с каждым из рассчитанных таким образом диапазонов, определяя комбинацию нажатых кнопок. (Полный текст программы не приводится).

Задача 3.5.4. Загадочный делитель напряжения (1 балл)

Делитель напряжения из двух резисторов разного номинала подключен между пинами D2 и D3, а средняя точка подключена к аналоговому пину A0, как показано на схеме:



Значение напряжения измеряется при разных комбинациях настроек пинов. Считать, что напряжение на пине в выходном режиме в точности равно либо 0, либо напряжению питания (хотя в реальных схемах, логический "0"всегда чуть выше 0V, а логическая "1"чуть ниже напряжения питания). Сопротивление внутреннего подтягивающего резистора (на любом пине) принять за 10 кОм.

Нужно написать программу, которая рассчитывает все возможные напряжения, которые можно намерить на 0000000000A0, изменяя настройки пинов. В этой задаче не используется мини-симулятор Ардуино, вы просто пишете код на любом удобном для вас языке.

Формат входных данных

Два числа через пробел - сопротивления резисторов R1 и R2.

Формат выходных данных

На первой строке - число значений в ответе, на второй строке - последовательность целых чисел в диапазоне 0..1023, которые можно было бы получить функцией analogRead(A0) при различных настройках пинов.

Все числа в ответе должны быть на одной строке через пробел, упорядочены по возрастанию, повторяющиеся значения удалены.

Допускается отклонение вычисленных значений на 1 в любую сторону.

Решение

Чтобы понять эту задачу, следует помнить, что любой пин Arduino может находится в 4-х разных состояниях:

Состояние	Обозн.	Что происходит
pinMode(pin, OUTPUT); digitalWrite(pin, HIGH);	1	На пин подано напряжение питания
pinMode(pin, OUTPUT); digitalWrite(pin, LOW);	0	На пин подается 0 (подключен GND)
pinMode(pin, INPUT);	I	Пин отключен от каких-либо напряжений
pinMode(pin, INPUT_PULLUP);	U	Пин подключен к напряжению питания через подтягивающий резистор в 10K

Все 4 перечисленных режима относятся как к пинам D2, D3, так и к пину A0, на котором производятся измерения. Когда мы говорим про измерение аналогового сигнала функцией analogRead(), всегда предполагается, что аналоговый пин находится в режимах INPUT или INPUT_PULLUP. Однако ничего не мешает установить A0 в режим OUTPUT, подать на него LOW или HIGH и прочитать значение полученного напряжения, если это зачем-то вдруг понадобилось.

Итак, мы можем получить $4 \cdot 4 \cdot 4 = 64$ комбинации состояний 3х пинов. Многие из них дадут уникальные комбинации сопротивлений и уникальные значения измеряемого напряжения на средней точке делителя. Следующая функция на Python вычисляет выходное напряжение при различных комбинациях состояний трех пинов (обозначены '0', '1', 'U', 'I' - см. выше):

Пример программы-решения

Ниже представлено решение на языке Python3

```
R1 = 4.7
R2 = 5.6
R_Int = 10.0

def find_value(A, D1, D2):
  # A0 - OUTPUT, полностью определяет результат
  if A == '0':
    return 0
  elif A == '1':
    return 1023

  if (D1 in '1U') and (D2 in '1U'):  # оба HIGH or PULLUP
    return 1023

  if D1 == '0' and D2 == '0':  # R1,R2 параллельно на GND
    if A == 'I':
      return 0  #  GND независимо от R1/R2
    Else:   #  R1, R2 параллельно, R_int сверху делителя
      r_dn = 1 / (1/R1 + 1/R2)
      r_up = R_Int
      return int(1023 * r_dn / (r_up + r_dn) )

  # Считаем верх делителя: начать с бесконечности, учитывать все
  # резисторы, идущие параллельно
  r_up = math.inf
  if A == 'U':
    r_up = 1 / (1/R_Int + 1/r_up)
  if D1 == '1':
```

```
    r_up = 1 / (1/R1 + 1/r_up)
  elif D1 == 'U':  # R_Int встает последов. с R1
    r_up = 1 / (1/(R1+R_Int) + 1/r_up)

  if D2 == '1':
    r_up = 1 / (1/R2 + 1/r_up)
  elif D2 == 'U':
    r_up = 1 / (1/(R2+R_Int) + 1/r_up)


  # Аналогично с "нижними" резисторами, но тут только могут быть R1 и R2
  r_dn = math.inf
  if D1 == 'O':
    r_dn = 1 / (1/R1 + 1/r_dn)
  if D2 == 'O':
    r_dn = 1 / (1/R2 + 1/r_dn)


  # Все 3 пина поставлены на INPUT, неопределенное напряжение.
  if r_up == math.inf and r_dn == math.inf:
    return None

  if r_dn == math.inf:
    return 1023
  if r_up == math.inf:
    return 0

  return int(1023 * r_dn / (r_up + r_dn) )


# Полный перебор состояний пинов
results = []
for A0 in "IUO1":
  for D1 in "IUO1":
      for D2 in "IUO1":
        v = find_value(A0, D1, D2)
        if not (v is None):
          results.append(v)


# Печатаем все уникальные решения, с сортировкой по возрастанию.
arr = [str(v) for v in sorted(set(results))]
print(len(arr))
print(' '.join(arr))
```

3.6. Шаговые двигатели и координатные устройства

Шаговые двигатели - основа самых разных координатных устройств, в том числе станков с ЧПУ, а среди них и самых распространенных - 3D-принтеров (да и обычных бумажных принтеров тоже!)

В отличие об обычных электромоторов, шаговый двигатель не просто "вращается а поворачивается на заданный угол, с весьма хорошей точностью, при подаче импульсов определенной формы, полярности и продолжительности на его 2 обмотки. Для управления шаговыми двигателями используют специальные ’ микросхемы-драйверы. Типичный драйвер управляется с микроконтроллера по 3-м пинам ENABLE, DIR и STEP:

ENABLE - включение двигателя. Когда шаговый двигатель включен, ток протекает по обмоткам и блокирует движение ротора, так что прокрутить вал можно, только приложив значительное усилие. Часто сигнал ENABLE для драйвера имеет инверсную полярность, т.е. двигатель включается при установке низкого уровня сигнала.

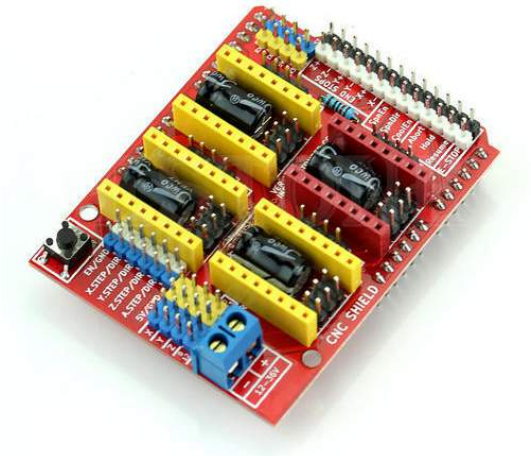
DIR - уровень сигнала определяет направление вращения.

STEP - каждый короткий импульс, подаваемый на этот пин, поворачивает вал шагового двигателя на 1 шаг.

Драйвер шагового двигателя несложно подключить к Ардуино прямо на макетной плате, но гораздо удобнее пользоваться специальными шилдами. На следующей картинке показан"бутерброд"из Arduino UNO (внизу), CNC ShieldV3 и 4-х драйверов моторов сверху. Это типичный расклад для любительского настольного фрезерного станка или плоттера. Для 3D-принтеров чаще применяют аналогичный "бутерброд но на базе Arduino Mega и шилда RAMPS.

Задача 3.6.1. Какие пины? ("Google it!") (1 балл)

Когда мы подключаем драйвер шаговых двигателей на макетной плате, то мы свободны в выборе пинов Arduino, которые будут им управлять. Но если мы используем CNC Shield, то вся "распиновка"определяется платой. Найдите в интернете информацию о CNC Shield V3 для Arduino UNO и заполните пробелы.



а) При использовании этой платой сигнал ENABLE:

(a) Отдельный для каждого мотора

(b) Общий для всех моторов

(c) Всегда включен

(d) Общий для каналов X и Y

б) Сигнал STEP для канала X подключен к пину Arduino:

(a) D1

(b) D2

(c) D3

(d) D4

(e) D5

(f) D6

(g) D7

(h) D8

в) Сигнал DIR для канала Z подключен к пину Arduino:

(a) D1

(b) D2

(c) D3

(d) D4

(e) D5

(f) D6

(g) D7

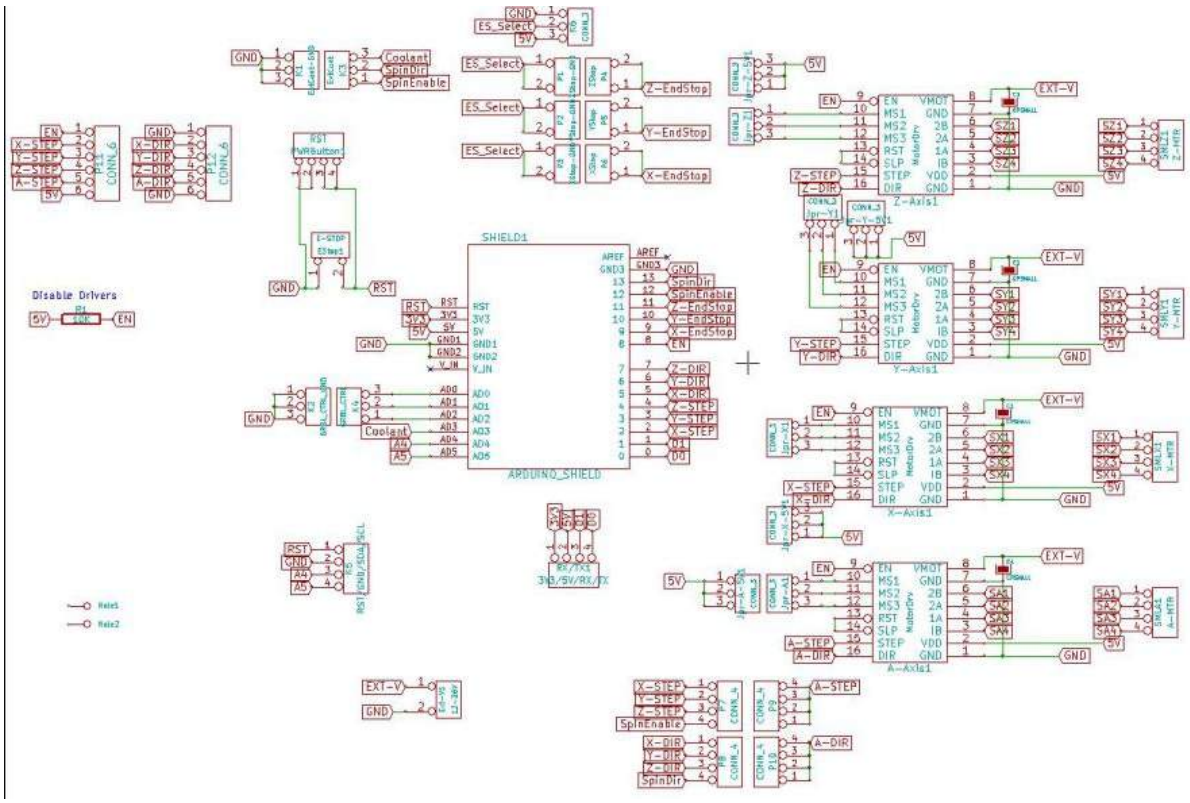
(h) D8

(i) D9

(j) D10

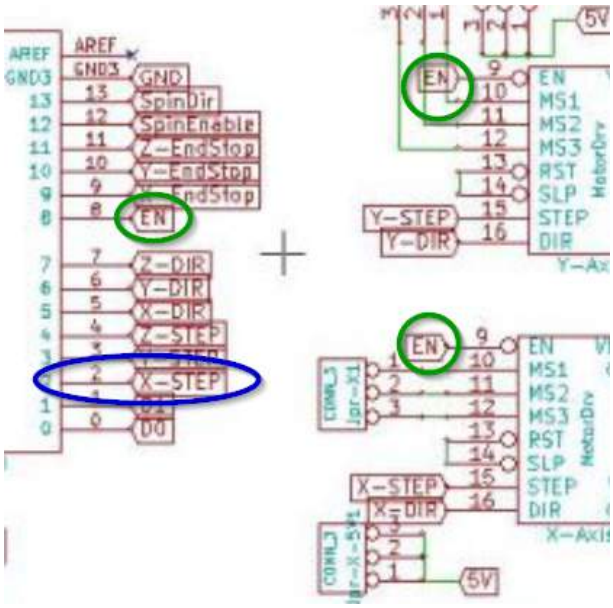
Решение

Решение этой задачи очевидно из условия. Ищем "CNC shield pinout", и находим вот такую схему:



Посмотрев подробнее на подключение драйверов, мы видим, что:

- Сигнал EN (а точнее - NOT ENABLE, т.к. драйверы включаются при подаче 0 на EN) разведен с пина 8 на все драйвера,
- Сигнал шаг X_STEP подключен к пину 2, Y_STEP к пину 3, Z_STEP к пину 4
- Сигнал направления X_DIR подключен к пину 5 и т.д.

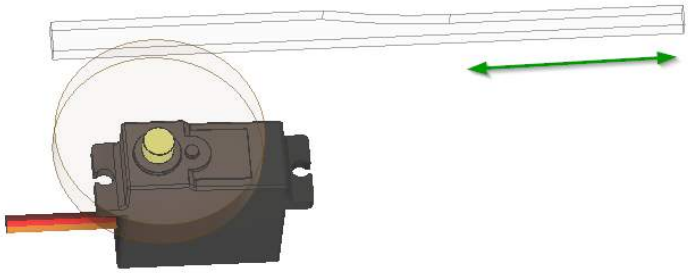


Этого достаточно, чтобы правильно ответить на вопрос задания. На практике, следует иметь в виду, что разводка конкретной "китайской" платы может отличаться от найденной вами в интернете, поэтому крайне желательно проверять ее по дорожкам на плате и/или мультиметром.

3.7. Механика

Задача 3.7.1. (5 баллов)

Вам надо спроектировать узел линейного перемещения для автомата, который раскладывает грузы по 8 ячейкам. Ячейки выстроены в одну линию, расстояние между соседними ячейками 40 мм. В качестве привода применен сервомотор с шестерней и зубчатой рейкой, как показано на рисунке:

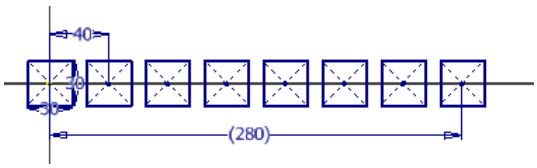


Сервопривод имеет вращательный момент (stall torque) 0.2 Н · м и диапазон поворота вала 270 градусов. Какое максимальное усилие можно получить на зубчатой рейке, при условии, что узел линейного перемещения должен обеспечивать позиционирование схвата над центром любой из ячеек?

Введите максимальное усилие, в Ньютонах (вводить только число), с точностью не ниже 0.1 Н.

Решение

Чем больше диаметр зубчатого колеса, тем больше ход зубчатой рейки, но меньше усилие. Нам нужно максимальное усилие, а значит, минимальный диаметр зубчатого колеса, при котором ход рейки оказывается достаточным для перемещения от центра первой до центра последней ячейки. Для N ячеек число "перегонов"между ячейками составит $N - 1$, а необходимый ход рейки $S = 7 \cdot 40 = 280$ мм:



Такой ход рейки должен составлять 3/4 окружности, поскольку угол поворота вала серво ограничен 270°. Следовательно, радиус зубчатого колеса будет равен:

$$R = \frac{4}{3} \cdot S/(2\pi) = \frac{2S}{3\pi} \approx 59.42 \text{ мм} \approx 0.0594 \text{ м}$$

Вращательный момент будет составлять:

$$T = \frac{Ts}{R} = \frac{0.2}{0.0594} = 3.37 \text{ Н}$$

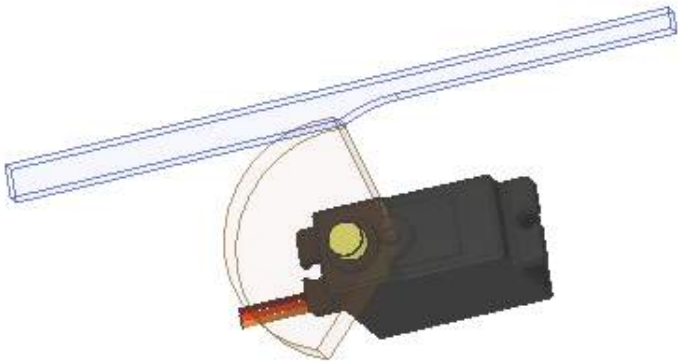
Обратите внимание, что в реальности зубчатое колесо не может иметь произвольный диаметр. При конструировании задается модуль зуба M, а диаметр делительной окружности зубчатого колеса получается умножением модуля зуба на число зубьев. Так, например, если было решено использовать зубчатую передачу с $M = 1.0$ мм, допустимые зубчатые колеса будут иметь целочисленный диаметр (а радиусы с шагом 0.5) и нам придется округлить результат до 60 мм.

В этом случае $T = 3.33$ Н. И тот и другой ответ находится в диапазоне допустимых погрешностей для данной задачи в Stepik.

Ответ: 3.33 ± 0.1

Задача 3.7.2. (4 балла)

Продолжаем конструировать тот же узел линейного перемещения. После покупки сервопривода неожиданно оказалось, что купленная модель обеспечивает поворот вала только на 120 градусов. Поэтому, и для сокращения размеров, было решено вместо полного зубчатого колеса использовать зубчатый сектор:



При использовании модуля зуба 1.5 мм, сколько зубьев будет иметь этот зубчатый сектор?

Ответ: 60 ± 1

Задача 3.7.3. (5 баллов)

Ось X 3D-принтера приводится в действие шаговым двигателем NEMA17 с вот такими параметрами:

На валу двигателя смонтирован шкив с 20 зубьями под зубчатый ремень типа GT2 (с шагом зуба 2 мм), перемещающий каретку. В драйвере мотора используется микрошаг 1/16.

Сколько шагов мотора необходимо выполнить, чтобы переместить каретку на A мм?

Решение

В данной задаче не требуется работать с силами и моментами вращения, но нужны базовые знания о принципах работы шагового двигателя и ременной передачи.

Из таблички с данными о моторе нам понадобится только одна величина: угол шага 1.8°. Это означает, что 1 оборот вал мотора совершит за $360/1.8 = 200$ шагов, или, наоборот, за 1 шаг вал поворачивается на 1/200 оборота. Кроме того, драйвер шагового двигателя умеет работать с микрошагами, искусственными промежуточными положениями вала внутри одного шага. Например, микрошаг 1/16 означает, что для поворота вала двигателя на один шаг нужно подать 16 управляющих импульсов.

Нам не потребуется диаметр шкива и число π, чтобы вычислять длину окружности. Достаточно знать, что шкив рассчитан на ремень с шагом зуба 2 мм и имеет 20 зубцов. Диаметр шкива подобран так, чтобы именно столько зубцов укладывалось по окружности. Это означает, что за один оборот шкива ремень продвинется на $S = 20 \cdot 2 = 40$ мм.

Таким образом, чтобы переместить каретку на X мм, на драйвер мотора необходимо подать N шаговых импульсов:

$$N = \frac{X}{40} \cdot 200 \cdot 16 = 80 \cdot X \text{ (микрошагов)}$$

В задаче в Stepik вместо переменной X каждый раз появляется случайное число, и вам нужно ввести числовой ответ.

Ответ: $80 \cdot a$

3.8. Практическое конструирование

Задача 3.8.1. (26 баллов)

Требуется сконструировать и изготовить П-образную мостовую конструкцию (мостовой кран), по верхней перекладине которой перемещается каретка, а в нижней части (у основания опоры) расположен мотор. Вам необходимо разработать механизм линейного перемещения каретки и способ передачи движения от мотора к каретке. Требование, чтобы мотор находился внизу конструкции (а не непосредственно на каретке, например), может показаться не слишком обоснованным, но оно введено специально для усложнения задачи.

Каретка может иметь произвольную форму, но должна обеспечивать перемещение вдоль перекладины с минимумом качаний и люфтов. Размер поперечного сечения перекладины и опор не должен превышать 40х40 мм. Высота конструкции 150-300 мм, ширина (между опорами) - от 200 до 400 мм. Диапазон перемещения каретки должен составлять не менее 80% расстояния между опорами.

В этом задании разрешается применить мотор любого типа, с любым способом управления (например, вручную переключателем), но лучше всего использовать шаговый двигатель, управляемый микроконтроллером.

Большая часть конструкции должна быть изготовлена с использованием технологий цифрового прототипирования (на ваш выбор и в соответствии с имеющимся оборудованием: 3D-печать, лазерная резка, фрезерование). Не разрешается использование конструктивных элементов из образовательных конструкторов (кроме мотора и шестерен, если требуется) или готовых узлов 3D-принтеров, но можно использовать трубки, профили, винтовые шпильки, ремни и т.п.

Представляемые результаты:

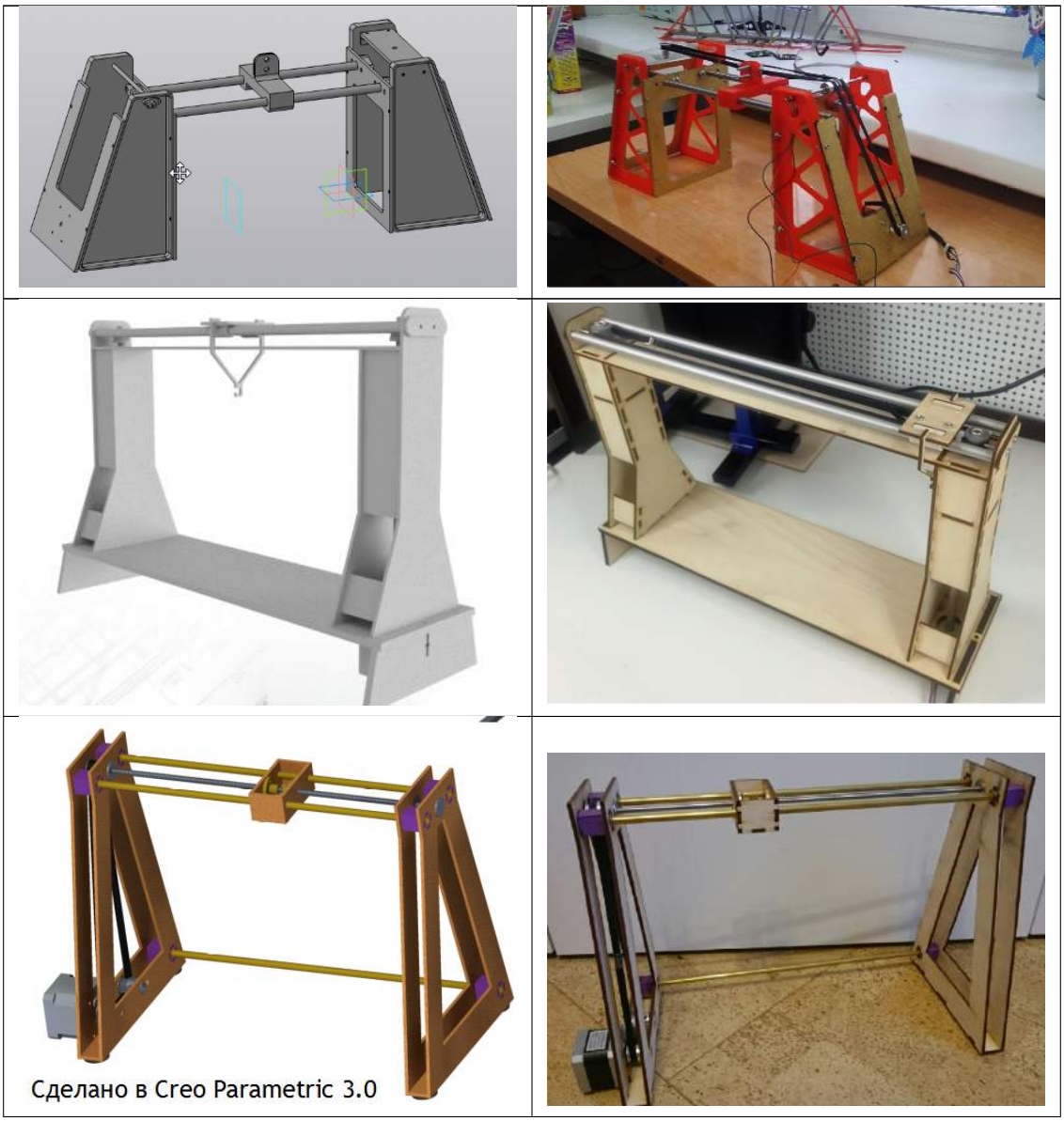
1. Сборочная модель конструкции, выполненная в любом САПР, с возможностью анимации
2. Видеоролик или презентация, показывающая основные шаги изготовления конструкции (3D-печать, лазерная резка, фрезерование, сборка и пр.)

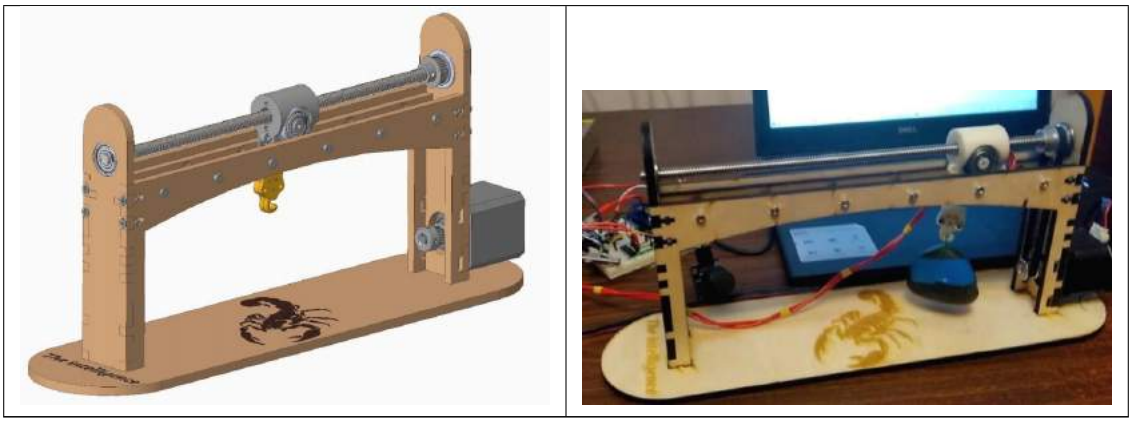
3. Видеоролик, показывающий работу конструкции.

При невозможности физически изготовить изделие, принимается анимированная сборочная модель (но, конечно, за нее начисляется меньше баллов). Учтите, что на очном туре команде все равно нужно как минимум уметь пользоваться 3D-принтером, а желательно еще и лазерным станком.

Решение

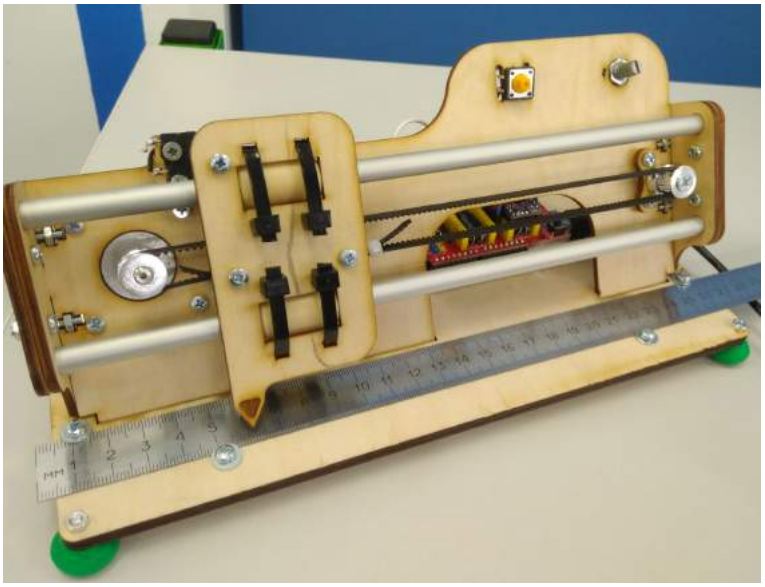
Это конструкторское задание, и для него нет единственно правильного решения. Вот какие изделия представили некоторые из участников 2-го тура, прошедшие в финал:





3.9. Практическая электроника (и еще немного конструирования)

Задача 3.9.1. Стенд с шаговым двигателем (12 баллов)



Изготовить простейший стенд для демонстрации линейного перемещения на основе шагового двигателя. Стенд должен состоять из шагового двигателя, двух шкивов, зубчатого ремня, направляющих, каретки, концевого выключателя и миллиметровой линейки. На каретке (или непосредственно на ремне) должен быть предусмотрен маркер, четко показывающий текущую позицию. Управление - от Arduino через драйвер шагового двигателя StepStick (на базе DRV8825, A4988 или аналогичных). Стенд должен демонстрировать: (а) калибровку - перемещение каретки в начальную позицию до замыкания концевого выключателя, при этом маркер должен находиться на нулевой позиции линейки, (б) В цикле, перемещение маркера к каждой из заданных точек, с кратковременной остановкой в каждой из этих точек. Точки задаются до начала выполнения теста произвольно, вразброс, значения крупно и разборчиво записываются на листе бумаги и предъявляются на видео перед началом теста.

Это задание на электронику и программирование, поэтому конструкция стенда

не оценивается (за исключением ситуаций, когда ужасное качество конструкции не дает возможность провести тест или оценить его результаты). Вам необязательно конструировать такую сравнительно "навороченную"конструкцию, как на фото выше, достаточно фанерки с закрепленным на ней двигателем и натяжным шкивом. Если данное задание и задание по конструированию выполняются командой одновременно, разрешается совместить их в одну конструкцию.

Рекомендуемый шаговый двигатель - любой NEMA 17 или NEMA 14.

Представляемые результаты:

- Текст программы управления шаговым двигателем
- Видеозапись работы устройства.

Решение

Это задание на электронику и программирование, поэтому конструкция стенда не оценивается (за исключением ситуаций, когда ужасное качество конструкции не дает возможность провести тест или оценить его результаты). Вам необязательно конструировать такую сравнительно "навороченную"конструкцию, как на фото выше, достаточно фанерки с закрепленным на ней двигателем и натяжным шкивом.

Если данное задание и задание по конструированию выполняются командой одновременно, разрешается совместить их в одну конструкцию. Большинство команд, прошедших во 2-й тур, так и поступили. Отдельные стенды представили только несколько команд, участники которых работали удаленно или где мостовой кран не был вовремя изготовлен.

Обратите внимание, что реально действующее координатное устройство обычно может позиционироваться любое число раз на любые (в рабочем диапазоне) заданные координаты. Поэтому в идеальном решении этого задания хочется видеть некую функцию, которая выполняет позиционирование, и эту функцию вы просто вызываете несколько раз в loop(), с задержками между вызовами. Соответственно, баллы за решение задачи могут снижаться, если:

- координаты в мм не задаются заранее, а получаются "какие получаются" экспериментально, в результате поворота вала мотора на не-пойми-какие углы
- логику программы пришлось бы сильно переделывать, чтобы обеспечить позиционирование произвольное число раз на произвольные углы
- программа неэффективно написана (есть очевидные улучшения, которые можно было бы применить, с существенным сокращением кода или улучшением читаемости/логичности)
- программа плохо отформатирована (хотя бы нажмите Ctrl-T, чтобы получить код "лесенкой", ставьте пустые строки между функциями, логически отдельными блоками определений переменных и т.п.)

Алгоритм калибровки: в цикле двигаем каретку по 1 шагу в направлении к концевому выключателю. По срабатыванию выключателя выходим из цикла и обнуляем переменную - счетчик позиции.

Алгоритм позиционирования: В цикле сравниваем счетчик текущей позиции с заданной позицией. Если текущая позиция не равна заданной, делаем шаг в нужную

сторону и соответственно изменяем на 1 текущую позицию. Когда текущая позиция выровнялась с заданной, выходим из цикла.

Полный код решения тривиален и здесь не приводится.

3.10. Программирование

Задача 3.10.1. Складской робот (5 баллов)

Имеется склад, представляющий собой прямоугольный массив ячеек. Размер массива задается входными данными. В ячейках могут размещаться грузы, опознаваемые по номерам. Все грузы уникальны (имеют разные номера), Имеется робот, который может перемещаться над ячейками, брать и переносить грузы. Робот умеет выполнять следующие команды: "G" (grab) - взять груз, "Mx,y" (move) - переместиться к ячейке с заданными индексами, "P" (put) - положить груз в ячейку. Индексы считаются от 0.

Напишите программу, расставляющую грузы на складе заданным образом.

Для упрощения примем, что ячейка, где должен находиться данный груз, не занята другим грузом (т.е. задача гарантированно решается за один проход, без перекладывания грузов в промежуточные ячейки). Оптимальность перемещений робота не учитывается, но избыточное число перемещений считается ошибкой.

Формат входных данных

На входе (по строкам):

N M - размеры склада

K - число грузов (всегда меньше числа ячеек N·M)

X Y T - K строк, описывающих начальное расположение грузов, где X,Y - координаты ячейки, T - числовой код груза

X Y T - еще K строк, описывающих требуемое конечное расположение грузов.

Формат выходных данных

Последовательность строк с командами для робота (G, P и Mx,y), по одной команде на строку. Полученная последовательность интерпретируется программой автопроверки. Ошибками считаются: неверная команда, попытка взятия груза из пустой ячейки, помещение груза в уже заполненную ячейку, попытка взять груз, когда робот уже несет груз, попытка выгрузить, когда груз не был взят, избыточное число команд, неверное размещение грузов после выполнения программы.

Стандартный ввод
3 2 2 0 0 2 2 1 1 0 1 1 2 0 2

Стандартный вывод
M2,1 G M0,1 P M0,0 G M2,0 P

Решение

Публикуемое ниже решение - одно из многих решений, присланных участниками 2-го тура. Решение прошло тесты и зачтено как правильное, хотя у автора задания есть подозрения, что правильность работы этой программы не гарантируется при изменении порядка следования записей (X Y T) во входных данных. Добавлены только комментарии:

Пример программы-решения

Ниже представлено решение на языке Python3

```
# Читаем данные из входного потока, как описано в задании
N, M = input().split()
N, M = int(N), int(M)
K = int(input())
begin = [[int(i) for i in input().split()] for i in range(K)]
end = [[int(i) for i in input().split()] for i in range(K)]

def sorting(a):  # функция сортировки по типу груза в ячейке
    return a[2]

begin.sort(key=sorting)
end.sort(key=sorting)

for i in range(K):
    # Не пытаемся переставлять, если та же позиция
    if begin[i][0] == end[i][0] and begin[i][1] == end[i][1]:
        continue
    print('M' + str(begin[i][0]) + ',' + str(begin[i][1]))
    print('G')
    print('M' + str(end[i][0]) + ',' + str(end[i][1]))
    print('P')
```