**Кейс 1. Система контроля параметров мусорных отходов.**

***Аннотация к кейсу****: Данный кейс посвящен ознакомлению с проблемой утилизации отходов человеческой жизнедеятельности и их влиянию на экологию, а также поиску решения данной проблемы через разработку системы сбора и вывоза мусора.*

***Уровень сложности кейса****: легкий/средний.*

***Оборудование, необходимое для кейса****: компьютер или ноутбук с выходом в интернет, среда программирования Arduino IDE, набор датчиков серии MQ, набор датчиков расстояния, устройства для световой и звуковой индикации, плата Arduino, монтажные провода.*

***Вид кейса:*** *репродуктивный.*

***Цель кейса:*** *разработка прототипа системы контроля параметров мусорных отходов для своевременного вывоза мусора.*

***Задачи кейса:***

* *Ознакомиться с проблемой своевременного вывоза мусора. Определить факторы, от которых зависит вызов устройств по вывозу мусора.*
* *Выбрать из предложенных датчиков те, которые могут детектировать опасные газы, выделяемые мусором, и уровень заполнения бака.*
* *Выбрать из предложенных устройств те, которые могут сигнализировать об опасном уровне концентрации вредных газов и заполненности бака .*
* *Собрать электрическую схему системы контроля мусорных отходов.*
* *Запрограммировать систему контроля параметров мусорных отходов.*

# *Проблемы, связанные с окружающей средой, очень актуальны на сегодняшний день. По мнению многих ученых, человечество уже живет в разрушающемся мире, в условиях все нарастающего экологического кризиса, который постепенно переходит в кризис всей цивилизации. Экологическим кризисом считается нарушения баланса в отношениях между человеком и природой.*

***Задание 1:*** *прочтите представленный ниже текст. Подумайте, как должна выглядеть система сбора мусора, какие параметры необходимо отслеживать внутри нее, для обеспечения безопасности окружающей среды.*

Москвичи и жители пригорода в год выбрасывают более 2,5 млн тонн отходов. Полигоны Московского региона завалены мусором, мусоросжигательных заводов не хватает, а те, что есть, устаревшие, не справляются с нагрузкой. Поэтому утилизация и переработка мусора в Московской области сегодня – огромная проблема.

Наполнение Подмосковных свалок составляет более 8 млн тонн в год (из них 5 млн привозятся из Москвы). Ежегодно количество мусора прирастает на 6%. Многие полигоны закрыты из-за переполненности, на некоторых отходы принимаются в ограниченном количестве. Поэтому растёт число мест стихийного сброса мусора (сегодня уже более 1200).

Большинство подмосковных полигонов сильно устарело, они переполнены и не справляются с таким количеством отходов. А это значит, что появляются вредные выбросы, неприятные запахи.

До 2013 года в Подмосковье действовало 39 полигонов по захоронению отходов, сейчас их осталось 15. При этом перерабатывается всего 7% мусора, все остальное – просто закапывается, из-за этого бывшие и действующие полигоны превратились в мусорные горы.

Создавать новые или расширять старые полигоны не выход, хотя сейчас именно это и происходит. Это чревато техногенными катастрофами вроде той, что произошла в подмосковном Волоколамске в 2018 году. На расположенную рядом с городом свалку «Ядрово» перенаправили потоки мусора с закрытых полигонов. Безопасное захоронение отходов требует целого ряда процедур вроде пересыпания землей и песком. Но когда мусора слишком много, правильная эксплуатация становится невозможна, а эффективность санитарных мер падает. В результате выделяется фильтрат, который проникает в грунтовые воды, отравляя все вокруг на многие километры. Еще опаснее выделение «свалочного газа», метана. Именно выброс этого газа в Волоколамске и привел к отравлению более 240 жителей города, включая 60 детей.

Метан горюч, поэтому на полигонах часто происходят возгорания, что усиливает загрязнение атмосферы. С 2019 года в России стартовала мусорная реформа. Но за 2,5 года добиться сколько-нибудь ощутимых перемен к лучшему не удалось. Одно из важных мероприятий для реализации этого – ввод раздельного сбора мусора на территории области. С этой целью разрабатывают двухконтейнерную систему: так, первый бокс будет для сухих отходов, второй – для мокрых. Подразумевается, что к сухим отходам относится то, что можно использовать повторно – это и полимеры, и бумага, и металл, и стекло. «Мокрый» бокс предназначен для пищевых и растительных отходов, т. е. для того, что выметают с улиц после уборки дворов.

**Информация для преподавателя:** *обсудите с участниками кейса прочитанный текст. Подведите их к правильному ответу:*

*Одним из способов реализации мусорной реформы является раздельный сбор мусора.* ***На какие типы можно разделить мусорные отходы?*** *На бумажные, стеклянные и смешанные отходы.* ***Какие параметры необходимо контролировать в каждом контейнере?*** *В контейнере с бумагой - возгораемость или дым, а также уровень его заполненности; В контейнере со стеклом - уровень заполненности; В контейнере со смешанными отходами - метан****,*** *возгораемость или дым, а также уровень его заполненности;*

***Задание 2:*** *Опираясь на информацию ниже, подберите необходимый набор датчиков, способных контролировать параметры процессов, происходящих внутри, которые сигнализируют о необходимости замены бака. При выборе компонентов используйте наиболее рациональную конструкцию, содержащую минимально необходимое и достаточное для решения кейса количество датчиков.*

**Информация для преподавателя:** *Выбор датчиков для системы необходимо производить, в соответствии с типом отходов, для которых предназначен бак (бумага; стекло; смешанные отходы). В данном кейсе разобран бак для смешанных отходов.*

**Технические средства для контроля загрязнений окружающей среды**

Датчики серии MQ позволяют определять содержания различных вредных веществ в атмосфере. Технические характеристики датчиков серии MQ приведены в табл. 1

**Таблица 1**

|  |  |  |
| --- | --- | --- |
| **Тип датчика** | **Вид загрязнения определяемый датчиком** | **Пределы измерений** |
| MQ 3 | пары спирта | 0,05 мг/л–10 мг/л |
| MQ 4 | метана | 200–10000 ppm |
| MQ 5 | пропана, бутана, метана и коксового газа | пропан: 200–10000 ppm  изобутан: 200–10000 ppm  метан: 200–10000 ppm |
| MQ 6 | пропана и бутана | Пропан: 200–10000 ppm  Изобутан: 200–10000 ppm |
| MQ 7 | угарный газ | 20–2000 ppm |
| MQ 8 | водорода | 100–10000 ppm |
| MQ 9 | угарного газа, метана и сжиженного углеводородного газа | Угарный газ: 20–2000 ppm  Метан: 500–10000 ppm  Сжиженные углеводородные газы: 500–10000 ppm |

# 

# 

# В газоанализаторы серии MQ встроен нагревательный элемент, который необходим для химической реакции: **во время работы сенсор будет горячим**. Внешний вид платы сенсора серии MQ представлен на рис. 1

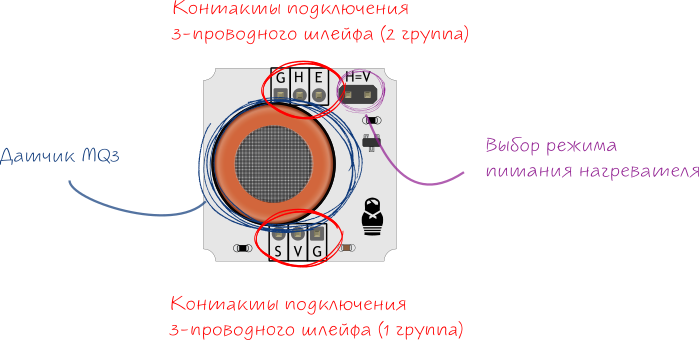
****

Рис. 1- Плата сенсора MQ

### 

# В датчиках серии MQ предусмотрена возможность контроля нагревательного элемента сенсора через отдельную группу контактов.

# Предусмотрено два режима работы, переключаемых джампером(рис.1):

# нагреватель датчика постоянно включён. В этом случае для подключения датчика необходим один трёх проводной шлейф(рис.2);

# управление нагревателем программно. В этом случае для подключения датчика необходим два трёх проводной шлейф (рис.3).

# 

# 

# 

# 

# 

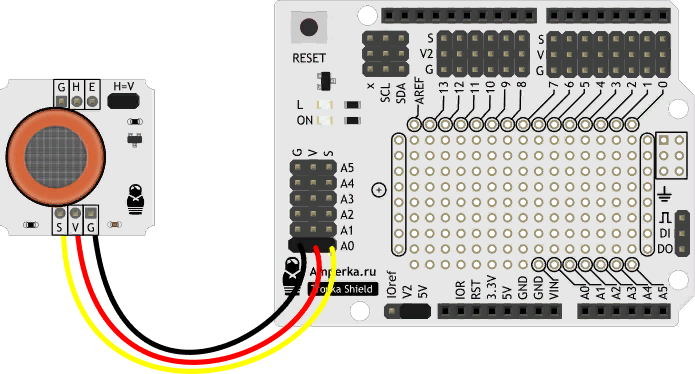


Рис. 2 - Постоянное включение нагревателя

# 

Рис.3 - Программное управление работой нагревателя

### Контакты подключения трёхпроводных шлейфов датчика серии MQ:

#### **1 группа**

# Сигнальный (S) — Выходной сигнал сенсора. Подключите к аналоговому входу микроконтроллера.

# Питание (V) — Питание датчика. Соедините с рабочим напряжением микроконтроллера.

# Земля (G) — Соедините с пином GND микроконтроллера.

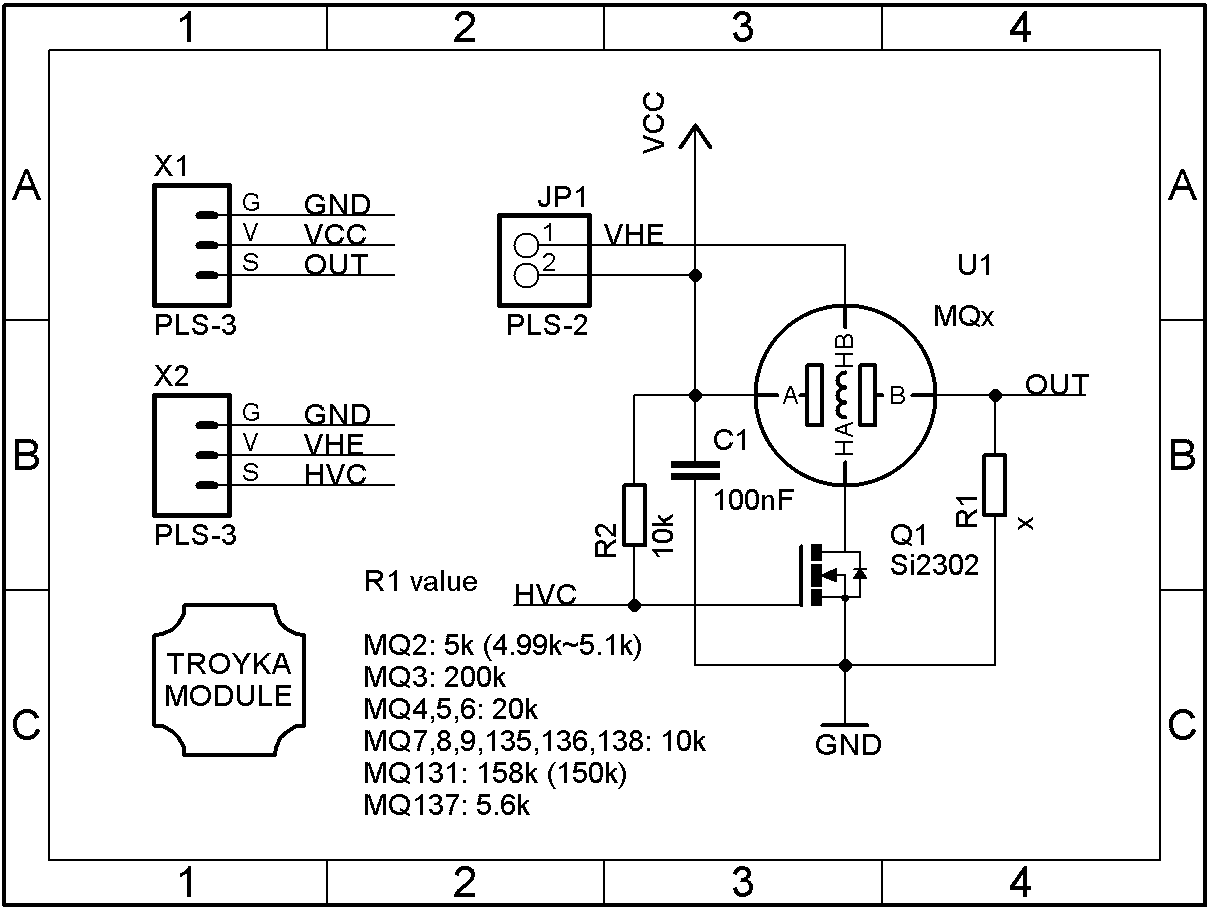
#### **2 группа**

# Сигнальный (E) — Управление питанием нагревателя. Подключите к цифровому пину микроконтроллера.

# Питание (H) — Питание нагревателя. Соедините с пином 5V

# Земля (G) — Соедините с пином GND микроконтроллера.

Принципиальная схема датчиков серии MQ приведена на рис. 4.

****

# Рис.4 - Принципиальная схема датчиков серии MQ

# Пример кода для обработки показаний датчика MQ3 представлен в листинге 1.

**Информация для преподавателя:** *Примеры листингов представлены для реализации разного уровня сложности кейса. Если выбранный вами уровень сложности* ***легкий****, прикладывать примеры листингов необязательно, так как в задании 5 будет готовый код с пояснениями. Если выбранный вами уровень сложности* ***средний****, то участники должны опираться на данные примеры, чтобы самостоятельно реализовать код программы в задании 5.*

Листинг 1

|  |
| --- |
| // библиотека для работы с датчиками MQ (Troyka-модуль)  #include <TroykaMQ.h>    // имя для пина, к которому подключен датчик  #define PIN\_MQ3 A0  // имя для пина, к которому подключен нагреватель датчика  #define PIN\_MQ3\_HEATER 13    // создаём объект для работы с датчиком  // и передаём ему номер пина выходного сигнала и нагревателя  MQ3 mq3(PIN\_MQ3, PIN\_MQ3\_HEATER);    void setup()  {  // открываем последовательный порт  **Serial**.begin(9600);  // включаем нагреватель  mq3.heaterPwrHigh();  **Serial.**println("Heated sensor");  }    void loop()  {  // если прошёл интервал нагрева датчика  // и калибровка не была совершена  if (!mq3.isCalibrated() && mq3.heatingCompleted()) {  // выполняем калибровку датчика на чистом воздухе  mq3.calibrate();  // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  **Serial**.print("Ro = ");  **Serial**.println(mq3.getRo());  }  // если прошёл интервал нагрева датчика  // и калибровка была совершена  if (mq3.isCalibrated() && mq3.heatingCompleted()) {  // выводим отношения текущего сопротивление датчика  // к сопротивлению датчика в чистом воздухе (Rs/Ro)  **Serial**.print("Ratio: ");  **Serial**.print(mq3.readRatio());  // выводим значения паров алкоголя  **Serial**.print(" Alcohol: ");  **Serial**.print(mq3.readAlcoholMgL());  **Serial**.print(" mG/L ");  **Serial**.print(mq3.readAlcoholPpm());  **Serial**.println(" ppm ");  delay(100);  }  } |

# Пример кода для обработки показаний датчика MQ4 представлен в листинге 2.

Листинг 2

|  |
| --- |
| // библиотека для работы с датчиками MQ (Troyka-модуль)  #include <TroykaMQ.h>    // имя для пина, к которому подключен датчик  #define PIN\_MQ4 A0  // имя для пина, к которому подключен нагреватель датчика  #define PIN\_MQ4\_HEATER 13    // создаём объект для работы с датчиком  // и передаём ему номер пина выходного сигнала и нагревателя  MQ4 mq4(PIN\_MQ4, PIN\_MQ4\_HEATER);    void setup()  {  // открываем последовательный порт  **Serial**.begin(9600);  // включаем нагреватель  mq4.heaterPwrHigh();  **Serial**.println("Heated sensor");  }    void loop()  {  // если прошёл интервал нагрева датчика  // и калибровка не была совершена  if (!mq4.isCalibrated() && mq4.heatingCompleted()) {  // выполняем калибровку датчика на чистом воздухе  mq4.calibrate();  // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  **Serial**.print("Ro = ");  **Serial**.println(mq4.getRo());  }  // если прошёл интервал нагрева датчика  // и калибровка была совершена  if (mq4.isCalibrated() && mq4.heatingCompleted()) {  // выводим отношения текущего сопротивление датчика  // к сопротивлению датчика в чистом воздухе (Rs/Ro)  **Serial**.print("Ratio: ");  **Serial**.print(mq4.readRatio());  // выводим значения газов в ppm  // выводим значения газов в ppm  **Serial**.print(" Methane: ");  **Serial**.print(mq4.readMethane());  **Serial**.println(" ppm ");  delay(100);  }  } |

# Пример кода для обработки показаний датчика MQ5 представлен в листинге 3.

Листинг 3

|  |
| --- |
| / библиотека для работы с датчиками MQ (Troyka-модуль)  #include <TroykaMQ.h>    // имя для пина, к которому подключен датчик  #define PIN\_MQ5 A0  // имя для пина, к которому подключен нагреватель датчика  #define PIN\_MQ5\_HEATER 13    // создаём объект для работы с датчиком  // и передаём ему номер пина выходного сигнала и нагревателя  MQ5 mq5(PIN\_MQ5, PIN\_MQ5\_HEATER);    void setup()  {  // открываем последовательный порт  **Serial**.begin(9600);  // включаем нагреватель  mq5.heaterPwrHigh();  **Serial**.println("Heated sensor");  }    void loop()  {  // если прошёл интервал нагрева датчика  // и калибровка не была совершена  if (!mq5.isCalibrated() && mq5.heatingCompleted()) {  // выполняем калибровку датчика на чистом воздухе  mq5.calibrate();  // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  **Serial**.print("Ro = ");  **Serial**.println(mq5.getRo());  }  // если прошёл интервал нагрева датчика  // и калибровка была совершена  if (mq5.isCalibrated() && mq5.heatingCompleted()) {  // выводим отношения текущего сопротивления датчика  // к сопротивлению датчика в чистом воздухе (Rs/Ro)  **Serial**.print("Ratio: ");  **Serial**.print(mq5.readRatio());  // выводим значения газов в ppm  **Serial**.print("LPG: ");  **Serial**.print(mq5.readLPG());  **Serial**.print(" ppm ");  **Serial**.print(" Methane: ");  **Serial**.print(mq5.readMethane());  **Serial**.println(" ppm ");  delay(100);  }  } |

# Пример кода для обработки показаний датчика MQ6 представлен в листинге 4.

Листинг 4

|  |
| --- |
| // библиотека для работы с датчиками MQ (Troyka-модуль)  #include <TroykaMQ.h>    // имя для пина, к которому подключен датчик  #define PIN\_MQ6 A0  // имя для пина, к которому подключен нагреватель датчика  #define PIN\_MQ6\_HEATER 13    // создаём объект для работы с датчиком  // и передаём ему номер пина выходного сигнала и нагревателя  MQ6 mq6(PIN\_MQ6, PIN\_MQ6\_HEATER);    void setup()  {  // открываем последовательный порт  **Serial**.begin(9600);  // включаем нагреватель  mq6.heaterPwrHigh();  **Serial**.println("Heated sensor");  }    void loop()  {  // если прошёл интервал нагрева датчика  // и калибровка не была совершена  if (!mq6.isCalibrated() && mq6.heatingCompleted()) {  // выполняем калибровку датчика на чистом воздухе  mq6.calibrate();  // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  **Serial**.print("Ro = ");  **Serial**.println(mq6.getRo());  }  // если прошёл интервал нагрева датчика  // и калибровка была совершена  if (mq6.isCalibrated() && mq6.heatingCompleted()) {  // выводим отношения текущего сопротивление датчика  // к сопротивлению датчика в чистом воздухе (Rs/Ro)  **Serial**.print("Ratio: ");  **Serial**.print(mq6.readRatio());  // выводим значения газов в ppm  **Serial**.print("LPG: ");  **Serial**.print(mq6.readLPG());  **Serial**.println(" ppm ");  delay(100);  }  } |

# 

# 

# Пример кода для обработки показаний датчика MQ7 представлен в листинге 5.

Листинг 5

|  |
| --- |
| // библиотека для работы с датчиками MQ (Troyka-модуль)  #include <TroykaMQ.h>    // имя для пина, к которому подключен датчик  #define PIN\_MQ7 A0  // имя для пина, к которому подключен нагреватель датчика  #define PIN\_MQ7\_HEATER 13    // создаём объект для работы с датчиком  // и передаём ему номер пина выходного сигнала и нагревателя  MQ7 mq7(PIN\_MQ7, PIN\_MQ7\_HEATER);    void setup()  {  // открываем последовательный порт  **Serial**.begin(9600);  // запускаем термоцикл  // в течении 60 секунд на нагревательный элемент подаётся 5 вольт  // в течении 90 секунд — 1,5 вольта  mq7.cycleHeat();  }    void loop()  {  // если прошёл интервал нагрева датчика  // и калибровка не была совершена  if (!mq7.isCalibrated() && mq7.atHeatCycleEnd()) {  // выполняем калибровку датчика на чистом воздухе  mq7.calibrate();  // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  **Serial**.print("Ro = ");  **Serial**.println(mq7.getRo());  // запускаем термоцикл  mq7.cycleHeat();  }  // если прошёл интервал нагрева датчика  // и калибровка была совершена  if (mq7.isCalibrated() && mq7.atHeatCycleEnd()) {  // выводим отношения текущего сопротивление датчика  // к сопротивлению датчика в чистом воздухе (Rs/Ro)  **Serial**.print("Ratio: ");  **Serial**.print(mq7.readRatio());  // выводим значения газов в ppm  **Serial**.print(" CarbonMonoxide: ");  **Serial**.print(mq7.readCarbonMonoxide());  **Serial**.println(" ppm ");  delay(100);  // запускаем термоцикл  mq7.cycleHeat();  }  } |

# Пример кода для обработки показаний датчика MQ8 представлен в листинге 6.

Листинг 6

|  |
| --- |
| // библиотека для работы с датчиками MQ (Troyka-модуль)#include <TroykaMQ.h>// имя для пина, к которому подключен датчик#define PIN\_MQ8 A0// имя для пина, к которому подключен нагреватель датчика#define PIN\_MQ8\_HEATER 13// создаём объект для работы с датчиком// и передаём ему номер пина выходного сигнала и нагревателяMQ8 mq8(PIN\_MQ8, PIN\_MQ8\_HEATER);void setup(){ // открываем последовательный порт  **Serial**.begin(9600); // включаем нагреватель mq8.heaterPwrHigh(); **Serial**.println("Heated sensor");}void loop(){ // если прошёл интервал нагрева датчика // и калибровка не была совершена if (!mq8.isCalibrated() && mq8.heatingCompleted()) { // выполняем калибровку датчика на чистом воздухе mq8.calibrate(); // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  **Serial**.print("Ro = ");  **Serial**.println(mq8.getRo()); } // если прошёл интервал нагрева датчика // и калибровка была совершена if (mq8.isCalibrated() && mq8.heatingCompleted()) { // выводим отношения текущего сопротивление датчика // к сопротивлению датчика в чистом воздухе (Rs/Ro)  **Serial**.print("Ratio: ");  **Serial**.print(mq8.readRatio()); // выводим значения газов в ppm  **Serial**.print("LPG: ");  **Serial**.print(mq8.readHydrogen());  **Serial**.println(" ppm "); delay(100); }} |

# 

# Пример кода для обработки показаний датчика MQ9 представлен в листинге 7.

Листинг 7

|  |
| --- |
| // библиотека для работы с датчиками MQ (Troyka-модуль)#include <TroykaMQ.h>// имя для пина, к которому подключен датчик#define PIN\_MQ9 A0// имя для пина, к которому подключен нагреватель датчика#define PIN\_MQ9\_HEATER 13// создаем объект для работы с датчиком// и передаем ему номер пина выходного сигнала и нагревателяMQ9 mq9(PIN\_MQ9, PIN\_MQ9\_HEATER);void setup(){ // открываем последовательный порт **Serial**.begin(9600); // запускаем термоцикл // в течении 60 секунд на нагревательный элемент подаётся 5 вольт // в течении 90 секунд — 1,5 вольта mq9.cycleHeat();}void loop(){ // если прошёл интервал нагрева датчика // и калибровка не была совершена if (!mq9.isCalibrated() && mq9.atHeatCycleEnd()) { // выполняем калибровку датчика на чистом воздухе mq9.calibrate(); // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт **Serial**.print("Ro = "); **Serial**.println(mq9.getRo()); // запускаем термоцикл mq9.cycleHeat(); } // если прошёл интервал нагрева датчика // и калибровка была совершена if (mq9.isCalibrated() && mq9.atHeatCycleEnd()) { // выводим отношения текущего сопротивление датчика // к сопротивлению датчика в чистом воздухе (Rs/Ro) **Serial**.print("Ratio: "); **Serial**.print(mq9.readRatio()); // выводим значения газов в ppm **Serial**.print(" LPG: "); **Serial**.print(mq9.readLPG()); **Serial**.print(" ppm "); **Serial**.print(" Methane: "); **Serial**.print(mq9.readMethane()); **Serial**.print(" ppm "); **Serial**.print(" CarbonMonoxide: "); **Serial**.print(mq9.readCarbonMonoxide()); **Serial**.println(" ppm "); delay(100); // запускаем термоцикл mq9.cycleHeat(); }} |

**Технические средства определения уровня наполнения бака**

Инфракрасный датчик расстояния Sharp GP2Y0A (рис.5) является популярным выбором для различных проектов на базе Arduino, которым требуется точное измерение расстояния.



Рис. 5- Инфракрасный датчик Sharp GP2Y0A

В датчиках Sharp установлен инфракрасный (IR) светодиод (LED) с линзой, который излучает узкий световой луч. Отраженный от объекта луч направляется через другую линзу на позиционно-чувствительный фотоэлемент (Position-Sensitive Detector, PSD) (рис.6). От местоположения падающего на PSD луча зависит его проводимость. Проводимость преобразуется в напряжение и, к примеру, оцифровывая его аналого-цифровым преобразователем микроконтроллера, можно вычислить расстояние.



Рис.6 - Схема измерения расстояния датчиком Sharp GP2Y0A

Выход инфракрасного датчика расстояния Sharp обратно пропорциональный — с увеличением расстояния его значение медленно уменьшается. Вид графика зависимости между расстоянием и напряжением приведен на рис.7.

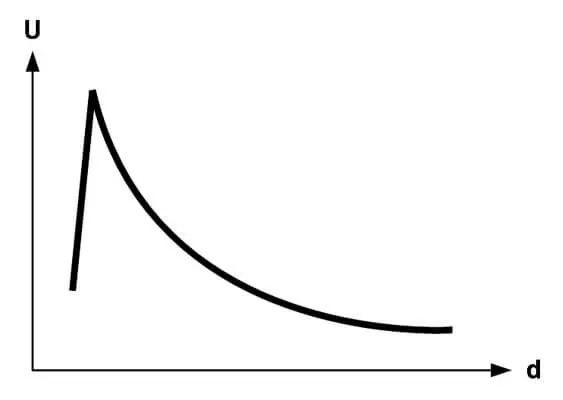


Рис. 7 Зависимость сигнала датчика от расстояния

Датчики, в зависимости от их типа, имеют границы измерения, в пределах которых их выход может быть признан надежным. Измерение максимального реального расстояния ограничивают два фактора: уменьшение интенсивности отраженного света и невозможность PSD регистрировать незначительные изменения местоположения отображенного луча. В целом график зависимости между расстоянием и напряжением не является линейным, однако в пределах допустимых расстояний график обратной величины выходного напряжения и расстояния к линейности приближается достаточно близко, и с его помощью довольно просто получить формулу для преобразования напряжения в расстояние. Для нахождения такой формулы необходимо точки этого графика ввести в какую-либо программу обработки табличных данных и из них создать новый график. В программе обработки табличных данных на основе точек графика возможно автоматически вычислить линию тренда. Например, для датчика GP2Y0A021YK0F графике представлен на рис. 8.

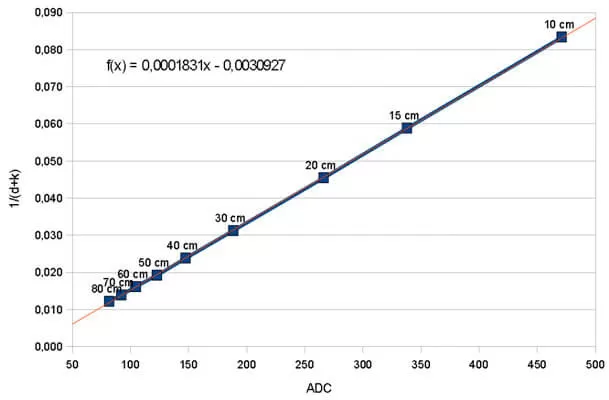


Рис.8 Линия тренда для датчика GP2Y0A021YK0F

Пример кода для обработки показаний с датчика GP2Y0A021YK0F приведен в листинге 8

Листинг 8

|  |
| --- |
| #define pin A0  //желтый провод дальномера подключаем к выводу А0 Arduino  void setup () {  **Serial**.begin (9600); // определяем скорость обмена данными с компьютером  pinMode (pin, INPUT);  }  void loop () {  float value = analogRead (pin); // считываем напряжение с дальномера  float range = get\_gp2d12 (value);  **Serial**.println (value);  **Serial**.print (range); // выдаем в порт расстояние  **Serial**.println (" mm");  **Serial**.println ();  delay (500);  }  float get\_gp2d12 (float value) { // переводим значение напряжения в расстояние  if (value < 10) value = 10;  return ((67870.0 / (value - 3.0)) - 40.0);  } |

**Ультразвуковой дальномер HC-SR04**

Ультразвуковой дальномер HC-SR04 предназначен для измерения расстояния от устройства до объекта. Работа модуля основана на принципе эхолокации. Модуль посылает ультразвуковой сигнал и принимает его отражение от объекта. Измерив время между отправкой и получением импульса, не сложно вычислить расстояние до препятствия. Поскольку в основе работы устройства используется ультразвук, модуль плохо подходит для определения расстояния до звукопоглощающих объектов. Для идеального измерения расстояния поверхность объекта должна быть ровной и гладкой. В этой статье мы подключим ультразвуковой дальномер HC-SR04 к Arduino.

Подключение ультразвукового датчика расстояния HC-SR04 к Arduino достаточно просто. Схема подключения показана на рис. 9.

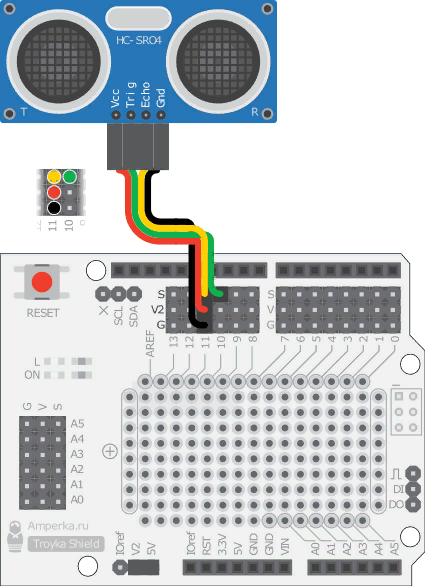


Рис. 9 - Подключение датчика HC-SR04

Контакт земли подключаем к выводу GND на плате Arduino, вывод питания соединяем с 5V. Trig и Echo подсоединяем к цифровым пинам платы.

Пример кода для обработки показаний с датчика HC-SR04 приведен в листинге 9.

Листинг 9

|  |
| --- |
| #define ECHO\_PIN 11  #define TRIG\_PIN 10    void setup() {  /\*задаем скорость общения с компьютером\*/  **Serial**.begin (9600);  /\* инициируем как выход TRIG пин \*/  pinMode(TRIG\_PIN, OUTPUT);  /\* инициируем как вход ECHO пин \*/  pinMode(ECHO\_PIN, INPUT);  }    void loop() {  int duration, cm;  /\* для большей точности установим значение LOW на пине Trig \*/  digitalWrite(TRIG\_PIN, LOW);  delayMicroseconds(2);  /\*установим высокий уровень на пине Trig \*/  digitalWrite(TRIG\_PIN, HIGH);  /\* Подождем 10 μs \*/  delayMicroseconds(10);  /\* установим низкий уровень на пине Trig \*/  digitalWrite(TRIG\_PIN, LOW);  /\* Узнаем длительность высокого сигнала на пине Echo \*/  duration = pulseIn(ECHO\_PIN, HIGH, 30000);  /\*Рассчитаем расстояние в сантиметрах \*/  cm = duration / 58;  /\* Отправляем полученные данные на компьютер\*/  **Serial**.print(cm);  **Serial**.println(" cm");  /\* Ждём четверть секунды и повторяем снова \*/  delay(250);  } |

***Задание 3:*** *опираясь на информацию ниже, подберите устройства, которые могут сигнализировать об опасном уровне концентрации вредных газов, заполненности бака. При выборе компонентов используйте наиболее рациональную конструкцию, содержащую минимально необходимое и достаточное для решения кейса количество устройств.*

**Технические средства для сигнализирование об опасном уровне концентрации вредных газов, заполненность контейнера.**

**Светодиод**

Для того, чтобы определить заполнен ли контейнер, на его внешней стенке необходимо установить световой индикатор. В нашем случае - это светодиод 5 мм Troyka - модуль (рис. 10).

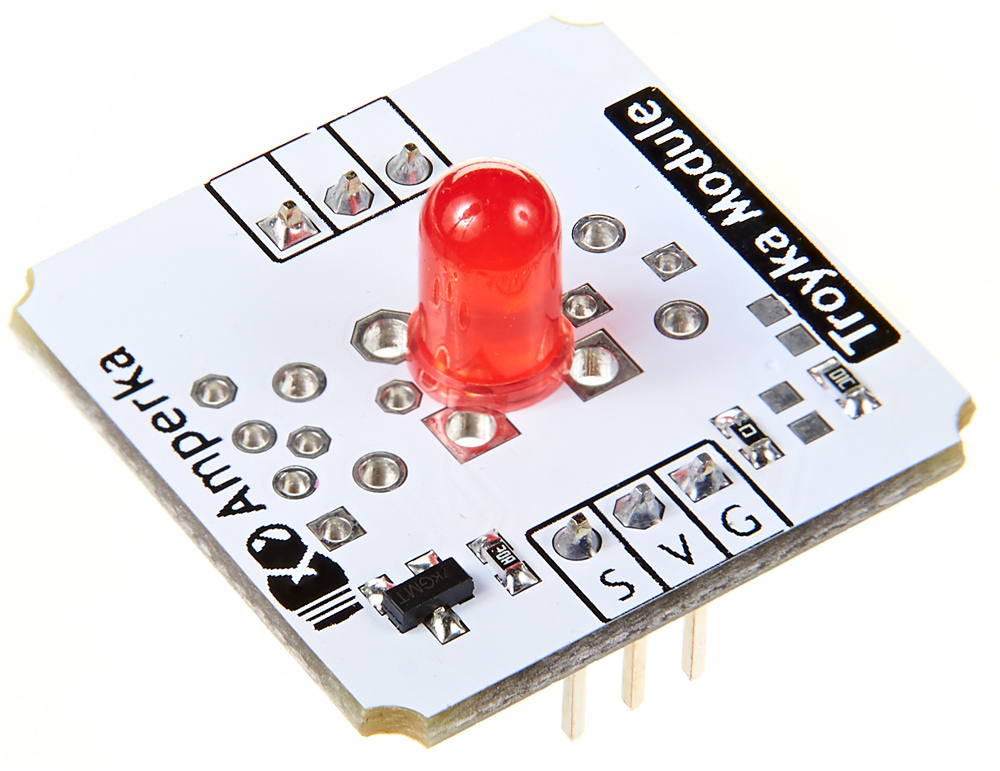
**

Рис. 10 - светодиод 5 мм Troyka - модуль

Светодиодный модуль имеет на борту транзистор, через который происходит управление светодиодом. Это означает, что модуль совместим как с сигналами на уровне 5 вольт, так и на уровне в 3,3 вольта. Пример кода по управлению светодиодом представлен в листинге 10.

Листинг 10

|  |
| --- |
| #define ledPin 2  void setup() {  pinMode(ledPin, OUTPUT);  }  void loop() {  digitalWrite(ledPin, HIGH);  delay(1000);  digitalWrite(ledPin, LOW);  delay(1000);  } |

**Зуммер**

Зуммер, пьезодинамик – все это названия одного устройства. Данные модули используются для звукового оповещения в тех устройствах и системах, для функционирования которых в обязательном порядке нужен звуковой сигнал. Широко распространены зуммеры в различной бытовой технике и игрушках, использующих электронные платы. Пьезопищалки преобразуют команды, основанные на двухбитной системе счисления 1 и 0, в звуковые сигналы.

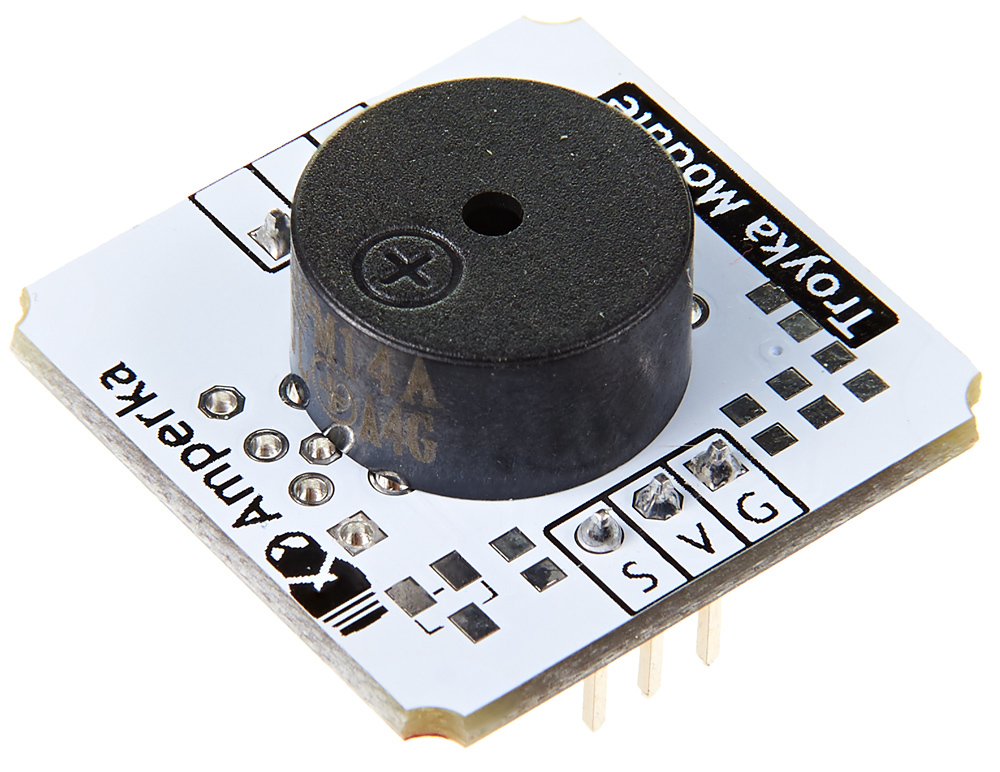
**

Рис. 11 - зуммер Troyka - модуль

Подключение модуля пьезоэлемента к Ардуино выглядит достаточно простым. Потребляемый ток маленький, поэтому можно просто напрямую соединить с нужным пином. Пример подключения модуля на рис. 12.

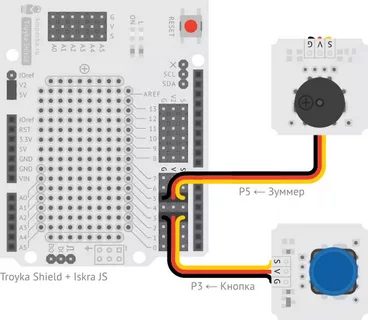
**

Рис. 12 - подключение зуммера к пину 5

Одним из простейших способов заставить заговорить пищалку является использование функции «analogwrite». Но лучше воспользоваться встроенными функциями. За запуск звукового оповещения отвечает функция «tone()», в скобках пользователю следует указывать параметры частоты звука и номера входа, а также времени. Для отключения звука используется функция «noTone()». Пример кода по управлению пьезодинамиком представлен в листинге 11.

Листинг 11

|  |
| --- |
| //Пин, к которому подключен пьезодинамик.  int piezoPin = 5;    void setup() {    }  void loop() {  /\*Функция принимает три аргумента  1) Номер пина  2) Частоту в герцах, определяющую высоту звука  3) Длительность в миллисекундах.  \*/  tone(piezoPin, 1000, 500); // Звук прекратится через 500 мс, о программа останавливаться не будет!  /\* Вариант без установленной длительности \*/  tone(piezoPin, 2000); // Запустили звучание  delay(500);  noTone(); // Остановили звучание  } |

***Задание 4:*** *соберите электрическую схему системы контроля мусорных отходов, используя устройства, которые вы выбрали в задании 2 и в задании 3.*

**Информация для преподавателя:** *Примеры подключений модулей необходимо дублировать в том случае, если выбранный вами уровень сложности кейса -* ***легкий****. Если уровень сложности -* ***средний****, участники должны выполнить данное задание самостоятельно.*

Выбранные Вами модули подключаются к управляющей электронике через 3-х проводные шилды. При подключении к [Arduino](https://amperka.ru/collection/arduino) будет крайне удобно использовать [Troyka Shield](https://amperka.ru/product/arduino-troyka-shield) (рис. 13). Подключения датчиков к [Troyka Shield](https://amperka.ru/product/arduino-troyka-shield) представлены на рис. 14-16.

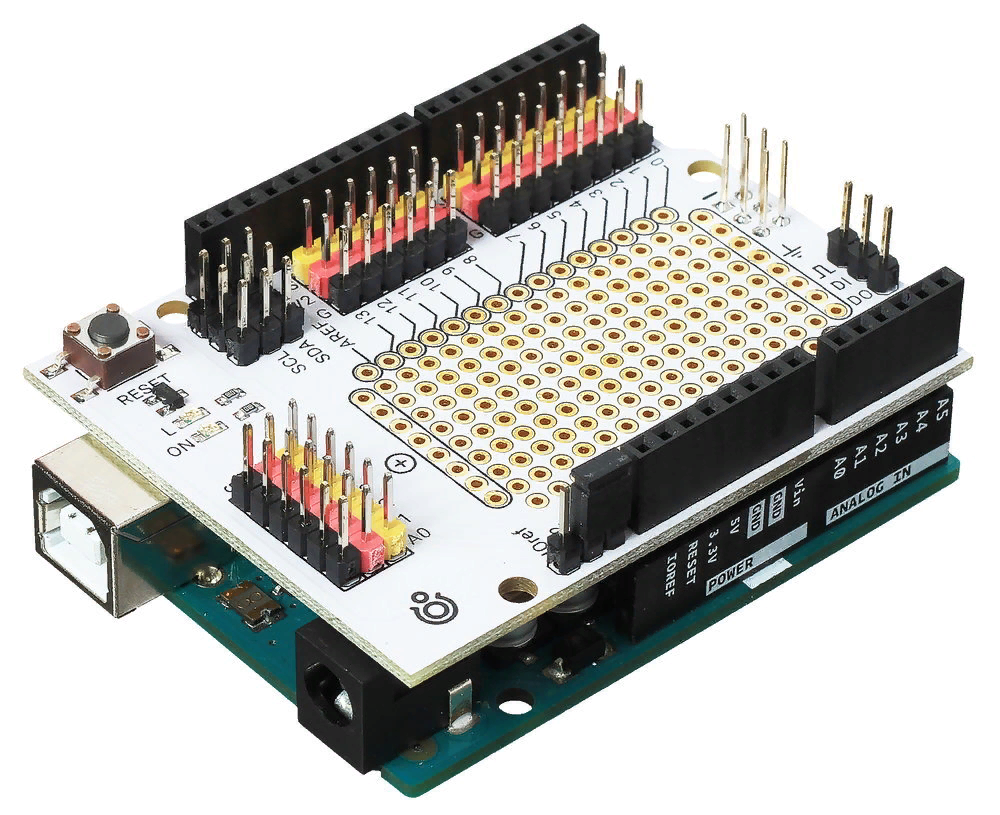
**

Рис. 13 - подключение [Troyka Shield](https://amperka.ru/product/arduino-troyka-shield) к [Arduino](https://amperka.ru/collection/arduino)

# 

Рис. 14 - подключение датчика газа к [Troyka Shield](https://amperka.ru/product/arduino-troyka-shield)

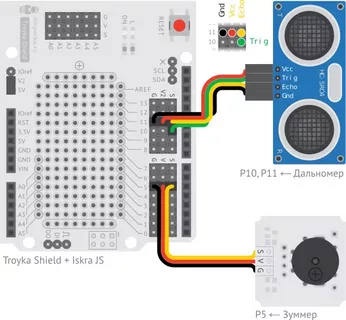


Рис. 15 - подключение ультразвукового дальномера и пьезодинамика к [Troyka Shield](https://amperka.ru/product/arduino-troyka-shield)

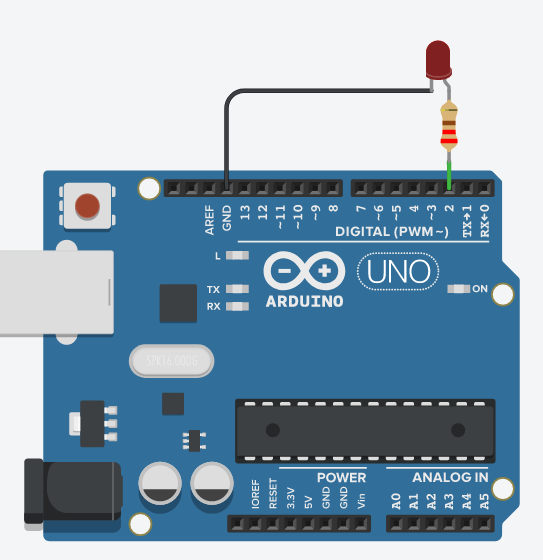


Рис. 16 - подключение светодиода к [Troyka Shield](https://amperka.ru/product/arduino-troyka-shield)

***Задание 5:*** *запрограммируйте систему контроля параметров мусорных отходов, произведите отладку программы, используя имитацию выделения опасных газов.*

***Важно:*** *так как в данном кейсе разрабатывается прототип системы, и в рамках урока нет возможности проверить ее в реальных условиях, создайте симуляцию опасных газов при помощи аромапалочки.*

**Информация для преподавателя:** *Для работы с датчиками газа необходимо подключить библиотеку TroykaMQ.h. Скачать ее можно по ссылке :* [*библиотека для Arduino*](https://github.com/amperka/TroykaMQ)*. Если выбранный вами уровень сложности кейса - легкий, то необходимо предоставить пример листинга 12. Если уровень сложности -* ***средний****, участники должны выполнить данное задание самостоятельно.*

**Добавление библиотеки для работы с датчиками газа**

Для того чтобы приступить к написанию кода, необходимо осуществить добавление библиотеки ***TroykaMQ.h.***

В среде программирование Arduino IDE выбирите вкладку “Скетч”, далее “подключить библиотеку” -> “Добавить .ZIP библиотеку...” (рис. 17). В открывшемся окне выберите папку или ZIP-архив с библиотекой, которую вы хотите добавить (рис. 18).

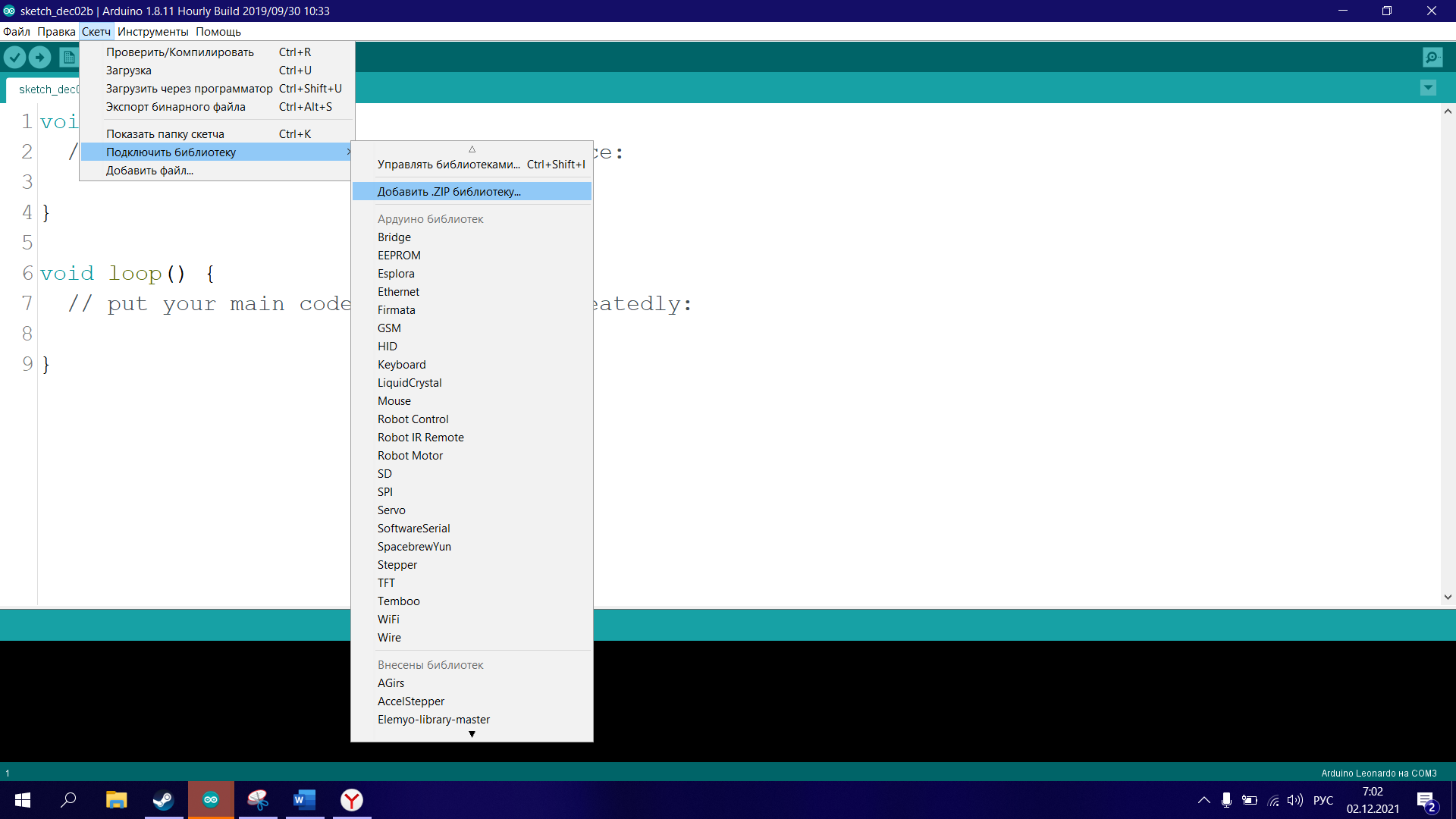
****

Рис. 17 - расположение функции “Добавить библиотеку”

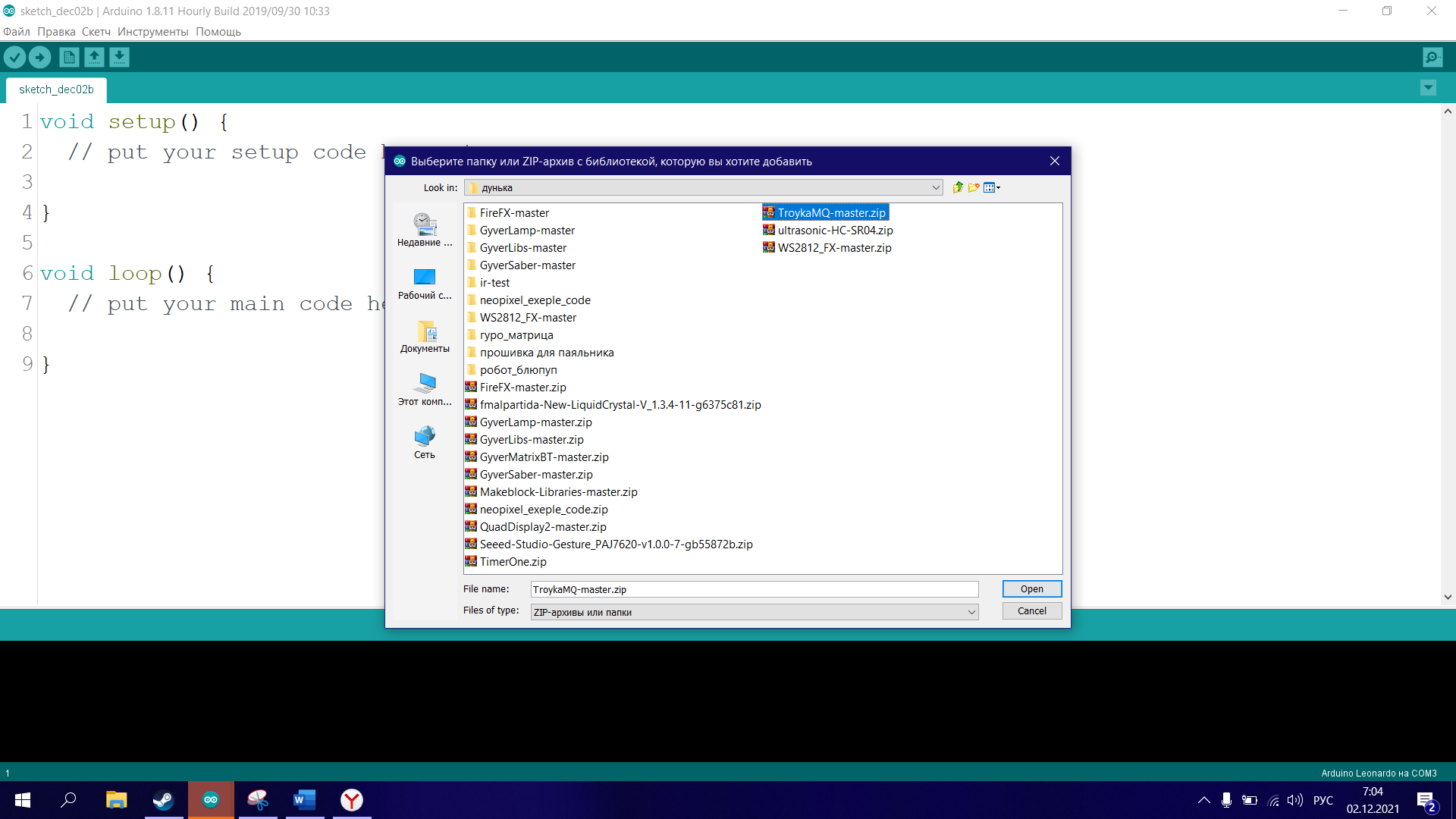
****

Рис. 18 - выбор архива с библиотекой

Реализуйте код для системы контроля параметров мусорных отходов. Данная программа должна выполнять следующие задачи:

1. Считывание параметров концентрации выбранного газа;
2. Считывание параметра с датчика расстояния;
3. Индикация превышения определенного уровня газа и заполненности бака.

**Информация для преподавателя:**

*При работе с датчиком газа, нужно подобрать пороговое значение опасной концентрации газа. Для этого необходимо написать программный код с выведением значений в последовательный монитор порт Arduino. Проверить значения исследуемого газа в чистой и загрязненной среде (поднести зажженную аромапалочку), пример выявления порогового значения на рис. 19. В качестве порогового значения следует взять усредненное между значением газа в чистой и загрязненной среде. Далее опираясь на найденное пороговое значение, написать условие срабатывания системы индикации.*

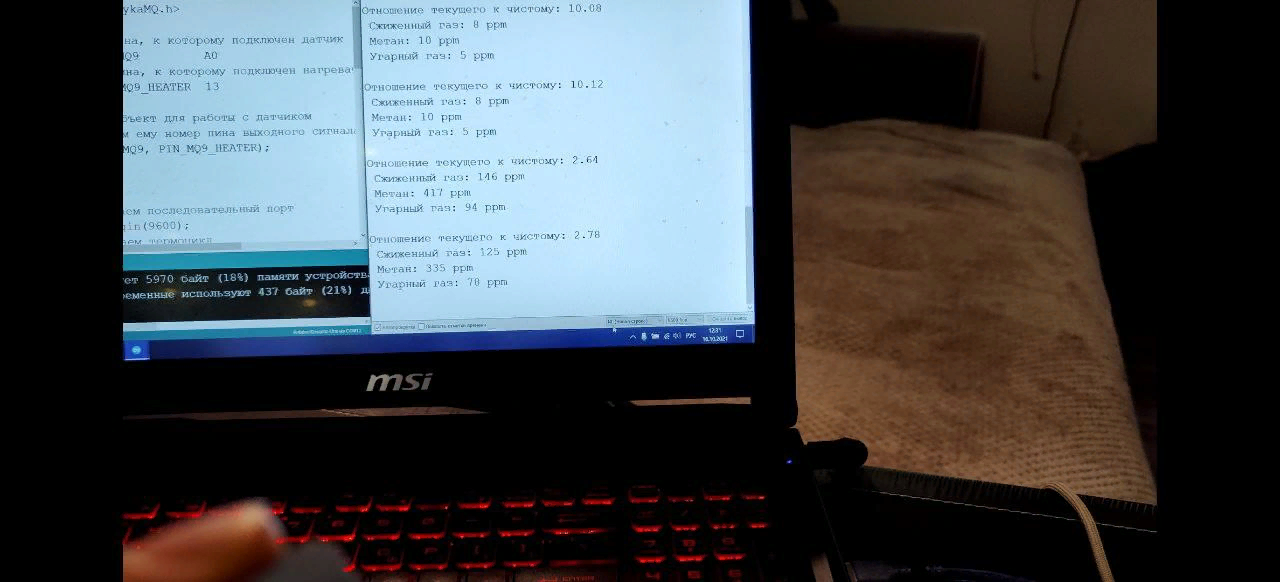


Рис.19

Листинг 12

|  |
| --- |
| #include <TroykaMQ.h>    // имя для пина, к которому подключен датчик  #define PIN\_MQ9 A0  // имя для пина, к которому подключен нагреватель датчика  #define PIN\_MQ9\_HEATER 13  //имя для пинов, к которым подключается ультразвуковой дальномер  #define PIN\_TRIG 12  #define PIN\_ECHO 11  //имя для пина, к которому подключена пьезодинамик  #define ZOOMER 9  //имя для пина, к которому подключен светодиод  #define Led 2  int duration, cm;  int Methane;  int CarbonMonoxide;    // создаём объект для работы с датчиком  // и передаём ему номер пина выходного сигнала и нагревателя  MQ9 mq9(PIN\_MQ9, PIN\_MQ9\_HEATER);    void setup()  {  // открываем последовательный порт  Serial.begin(9600);  // запускаем термоцикл  // в течении 60 секунд на нагревательный элемент подаётся 5 вольт  // в течении 90 секунд — 1,5 вольта  mq9.cycleHeat();  pinMode(PIN\_TRIG, OUTPUT);  pinMode(PIN\_ECHO, INPUT);  pinMode(ZOOMER,OUTPUT);  pinMode(Led,OUTPUT);  }    void loop()  {  ////////\*Работа с датчиком газов\*//////////  // если прошёл интервал нагрева датчика  // и калибровка не была совершена    if (!mq9.isCalibrated() && mq9.atHeatCycleEnd()) {  // выполняем калибровку датчика на чистом воздухе  mq9.calibrate();  // выводим сопротивление датчика в чистом воздухе (Ro) в serial-порт  Serial.print("Сопротивление датчика в чистом воздухе = ");  Serial.println(mq9.getRo());  // запускаем термоцикл  mq9.cycleHeat();  }  // если прошёл интевал нагрева датчика  // и калибровка была совершена  if (mq9.isCalibrated() && mq9.atHeatCycleEnd()) {  // выводим отношения текущего сопротивление датчика  // к сопротивлению датчика в чистом воздухе (Rs/Ro)  Serial.print("Отношение текущего к чистому: ");  Serial.println(mq9.readRatio());  // выводим значения газов в ppm  Serial.println(" ppm ");  Serial.print(" Метан: ");  Serial.print(mq9.readMethane());  Methane=mq9.readMethane();  Serial.println(" ppm ");  Serial.print(" Угарный газ: ");  Serial.print(mq9.readCarbonMonoxide());  CarbonMonoxide=mq9.readCarbonMonoxide();  Serial.println(" ppm ");  Serial.println(" ");  delay(100);  if(CarbonMonoxide>100||Methane>100){// если один из параметров станет больше 100 ( этот параметр подбирается для каждого индивидуально,  //в процессе тестирования своей системы), то необходимо подать звуковой сигнал  tone(9, 1000);  delay(100);  }  else{  noTone(9);  delay(100);  }  // запускаем термоцикл  mq9.cycleHeat();  }  ////////////////\* работа с ультразвуковым дальномером\*///////////////  // Сначала генерируем короткий импульс длительностью 2-5 микросекунд.  digitalWrite(PIN\_TRIG, LOW);  delayMicroseconds(5);  digitalWrite(PIN\_TRIG, HIGH);  // Выставив высокий уровень сигнала, ждем около 10 микросекунд. В этот момент датчик будет посылать сигналы с частотой 40 КГц.  delayMicroseconds(10);  digitalWrite(PIN\_TRIG, LOW);  // Время задержки акустического сигнала на эхолокаторе.  duration = pulseIn(PIN\_ECHO, HIGH);  // Теперь осталось преобразовать время в расстояние  cm = (duration / 2) / 29.1;  Serial.print("Расстояние до объекта: ");  Serial.print(cm);  Serial.println(" см.");  if (cm<5){  digitalWrite(Led, HIGH); // Включаем светодиод  delay(10);  }  else{  digitalWrite(Led, LOW); // Выключаем светодиод  delay(10);  }  // Задержка между измерениями для корректной работы скеча  delay(250);  } |