5. Скачиваем программу Putty – это программа для чтения данных с портов. Плюс её в том, что она поддерживает вывод пояснений на русском языке.

6. Запускаем программу.

7. Подключите вашу плату Arduino к USB - разъему компьютера.

8. В меню Инструменты программы выбираем Плата. Наша плата Arduino UNO Там же выбираем COM port к которому подключена плата.

Теперь наш компьютер готов к работе как с нашим макетным стендом, так и с любыми системами, созданными на базе Arduino-совместимых продуктов.

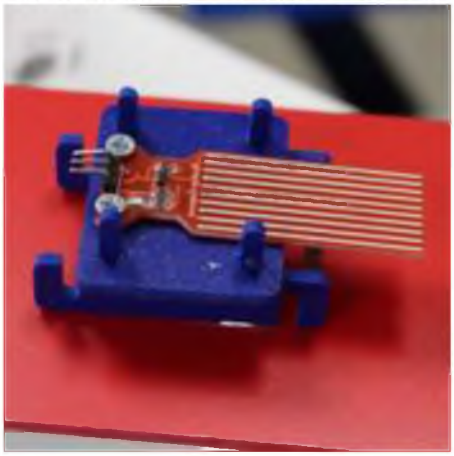
Примеры моделирования ситуаций.

1. Протечка

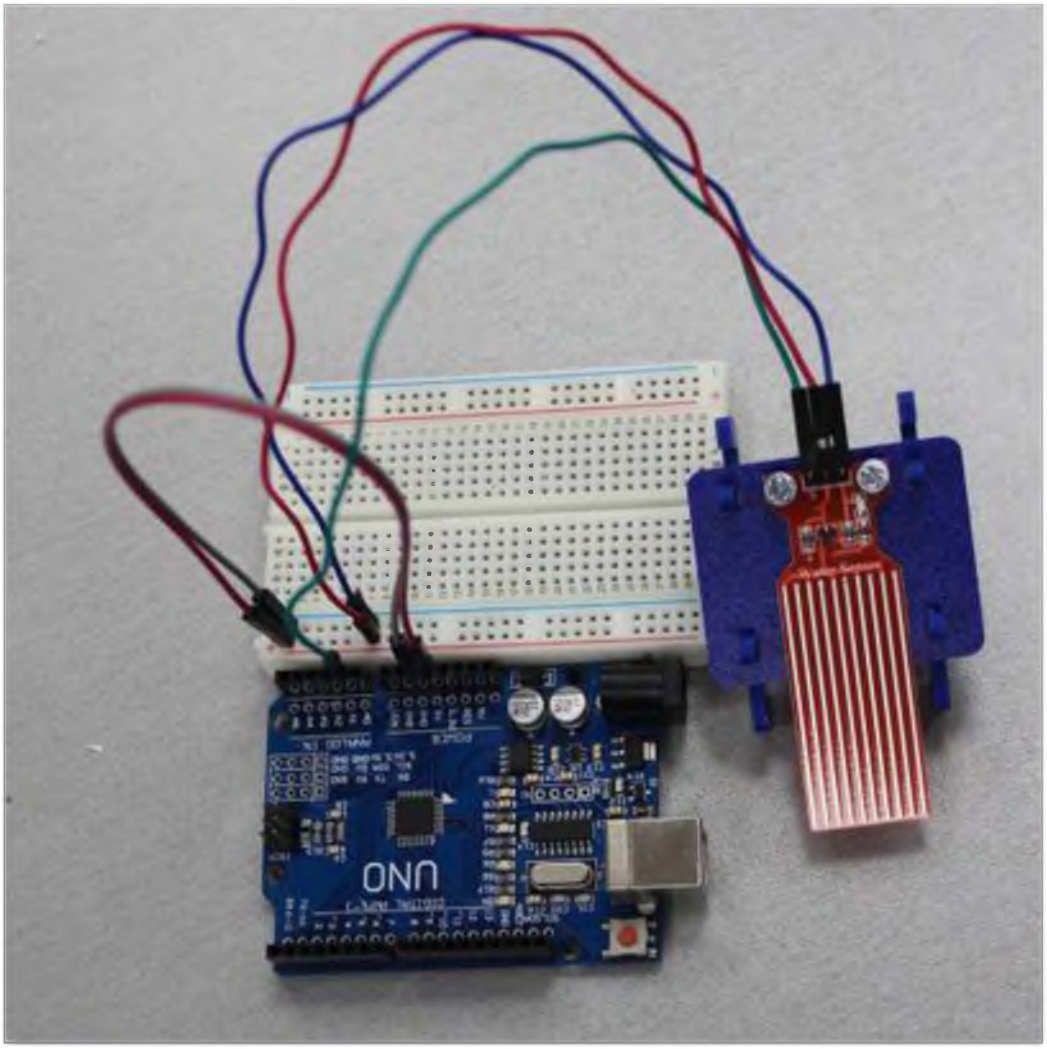
Для моделирования этой ситуации нам потребуются всего один датчик и контроллер (в нашем случае это Arduino UNO). Схема работы датчика протечек предельно проста. На подложке из текстолита расположены дорожки. Как известно, вода, если она не дистилированная, прекрасно проводит электрический ток за счет растворенных в ней разичных солей. На этом и основан принцип работы даного датчика. При появлении жидкости, контакты (дорожки) на его поверхности перемыкаются и на выход датчика поступает сигнал о протечке.

Сам датчик выглядит так, как на фото:

При установке такого датчика на крыше, он может быть использован как датчик дождя.



В нашем случае коммутация (для всех датчиков) осуществляется следующим образом. Сигнальный контакт датчика мы соединяем проводом с соответствующим контактом контроллера (с тем, который укажем в дальнейшем в программе), а контакты, обеспечивающие питание, мы соединяем с шиной питания на макетной плате (см. рисунок). Питание 5В мы подаем на шину макетной платы с контроллера.



Программа, написанная в среде Arduino, называется скетч. Рассмотрим скетч для датчика протечек:

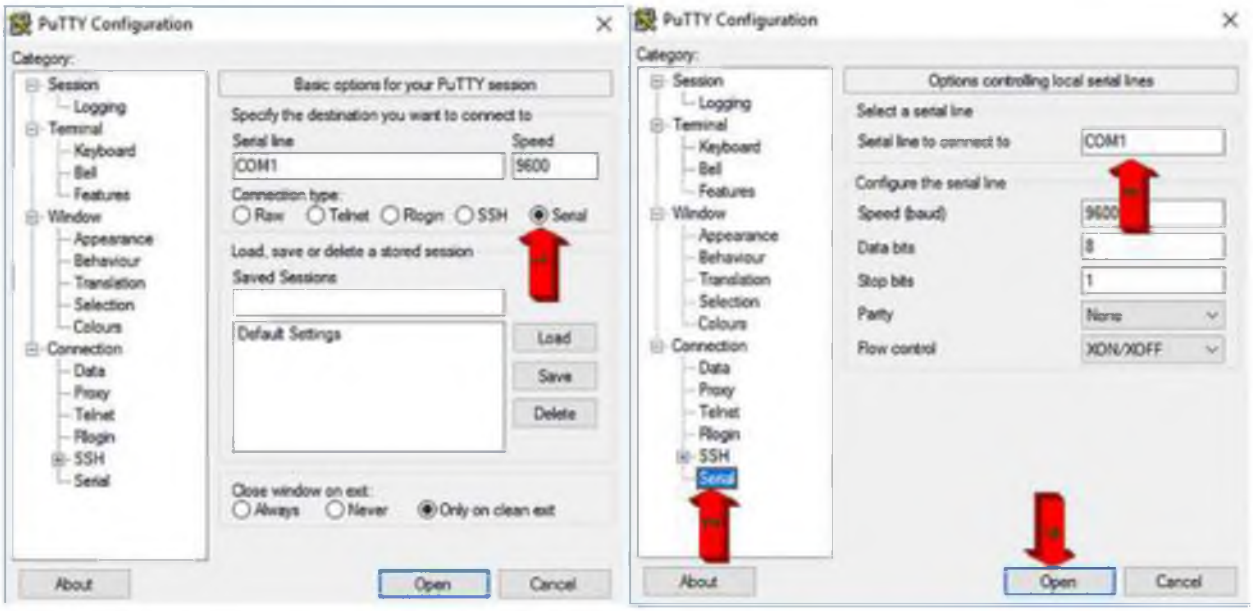
Датчик Протечек (Уровня воды)

```
int waterPin = 0; // Задаем переменную для значения наличия воды и ее начальное значение 0 (сухо)
{
 Serial.begin(9600); // Инициализируем передачу данных в COM-порт со скоростью 9600
}
void loop() // Открываем цикл в котором будем опрашивать датчик каждые 1000 милисекунд
{
waterPin = analogRead(A1); // Считываем данные с датчика на контакт A1 контроллера и
                           // присваеваем его нашей переменной waterPin
Serial.println(waterPin); // Выводим значение переменной waterPin в COM-порт
Serial.print("Наличие воды    "); // Добавляем к значению переменной надпись «Наличие воды»

delay(1000); // Ждём 1000 милисекунд до следующего опроса датчика
}
```

Датчик протечек (уровня воды) аналоговый и может быть установлен в ёмкость с водой для измерения ее уровня. При этом отслеживается изменение сопротивления между контактами и на выход подаются различные значения, отличные от ноля.

Запустим среду Arduino IDE и внесем в нее этот скетч. Подключаем наш контроллер к компьютеру с помощью USB-кабеля (входит в комплект поставки) и в меню «Инструменты» выбираем тип платы Arduino и COM-порт к которому она подключена. Сверху в окне программы жмем галочку – запустится проверка скетча. Если проверка скетча завершилась успешно, то нажимаем в меню программы стрелочку для загрузки скетча в контроллер. После того, как появится надпись «Загрузка завершена», наша схема готова к работе.
Что бы вывести на экран данные, выдаваемые скетчем в COM-порт, нам понадобится програма Putti, которую мы установили ранее. Для работы программы нам нужно правильно выставить в ней режим работы – в нашем случае COM и номер порта Arduino (тот же, что в среде разработки) на соответствующей вкладке:



После этого нажимаем в Putty кнопку «Open» - откроется окно в котором появится результат работы нашего скетча.

2. Незаконное проникновение или препятствие по курсу

Для моделирования этих ситуаций мы используем ультразвуковой датчик расстояния.

У этого датчика есть несколько особенностей на которых необходимо остановиться отдельно.

- Используются не 3, а 4 контакта из которых два отвечают за питание, один за излучение сигнала и один за прием сигнала.
- Значение расстояния, получаемое с датчика требует пересчета по формуле для получения истинного значения.

Датчик требует для работы установки библиотеки «Ultrasonic».

Подключение датчика к контроллеру и питанию осуществляется аналогично рассмотренному выше датчику протечек, но на контроллере задействуются 2 контакта.

Рассмотрим скетч для ультразвукового датчика расстояния:

```
#include <Ultrasonic.h> // подключение библиотеки
int echoPin = 11;  // приемник сигнала подключен к цифровому контакту 11 на Arduino
int trigPin = 12; // излучатель сигнала подключен к цифровому контакту 12 на Arduino
void setup()
{
  Serial.begin (9600); // инициализируем COM-порт
  pinMode(trigPin, OUTPUT); // определяем режим для trigPin
  pinMode(echoPin, INPUT);  // определяем режим для echoPin
}
void loop()  // все измерения и вывод значений выполняем в цикле
{
  int duration, cm;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  cm = duration / 58; // пересчет показаний датчика в сантиметры
  Serial.print (cm);
  Serial.println (" cm");
  delay(100);
}
```

В результате работы данного скетча, на экране в Putty мы можем увидеть значение расстояния до объекта в сантиметрах.

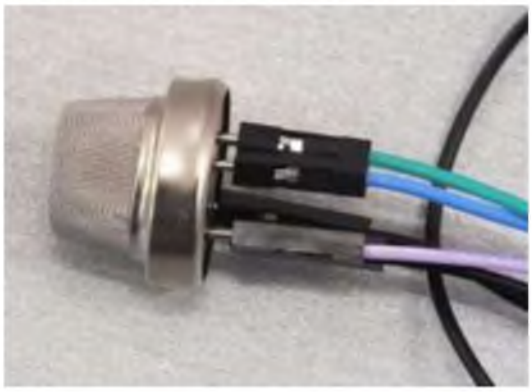
Если мы используем наш датчик расстояния на подводной лодке, то он отвечает за определения наличия препятствия по курсу и просчета расстояния до него.

В системах умного дома этот же датчик используется для определения незаконного проникновения (один или совместно с датчиками движения и шума). В этом случае инициализация сигнала о ситуации происходит при любом изменении расстояния в зоне действия датчика (например, открытие окна или двери).

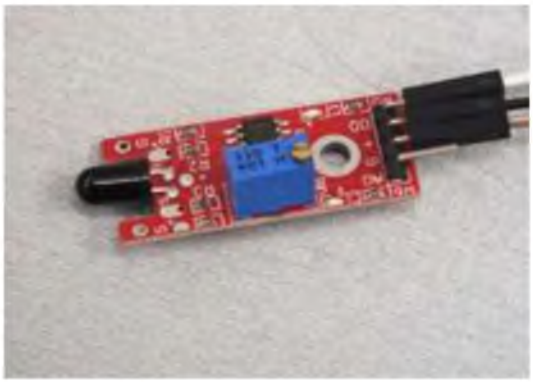
3. Пожар

Для моделирования ситуации сначала давайте разберемся какие факторы нам могут указать на наличие пожара. Это наличие открытого огня, задымление и повышение температуры окружающей среды. Всегда-ли наличие открытого огня указывает на пожар? Далеко нет. Огонь может быть от зажигалки, от кухонной плиты или от чего-то еще, что не представляет непосредственной опасности помещению. Тоже самое в отдельности можно сказать и о задымлении, которое может возникнуть от дымовой шашки или дым может быть принесен потоком воздуха и о повышении температуры, когда комната может быть

нагрета солнцем через окно. Таким образом, на пожар может указывать наличие минимум двух из перечисленных факторов. Возьмем это определение за основу при работе над моделированием ситуации. Исходя из выше описанного, возьмем три датчика:
- датчик широкого спектра газов MQ2 для анализа задымленности;

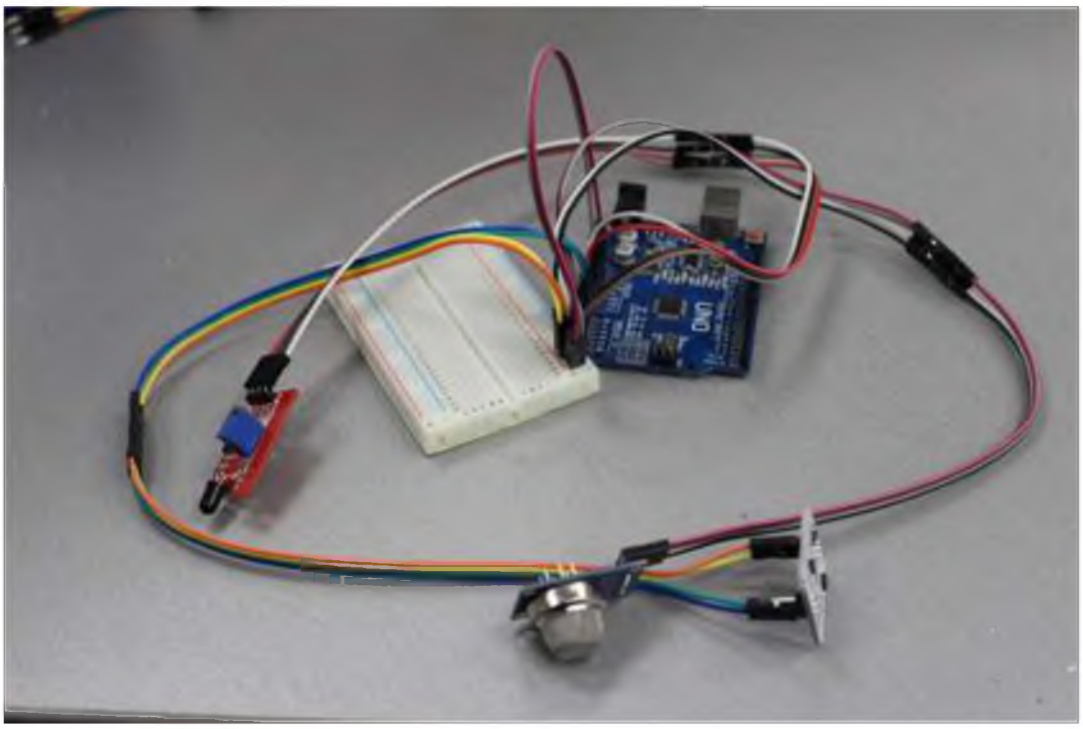


- датчик пламени (срабатывает при наличии открытого огня);



- датчик температуры (в данном примере используем барометр, который снимает значения как давления, так и температуры)

Каждый из этих датчиков следит за своими параметрами среды в помещении.
Но простого подключения датчиков мало – при срабатывании системы нам должен поступать сигнал о чрезвычайной ситуации. В простейшем варианте это может быть звуковой или световой сигнал, а в более сложных передача информации о пожаре на пульт пожарным или на телефон владельца помещения.
Контакты, отвечающие за снятие показаний с датчиков, мы подключаем мы подключаем к контроллеру (в нашем случае это Arduino UNO), а контакты питания выводим на макетную плату. Питание 5В для датчиковмы можем взять с платы контроллера или с отдельного источника питания. Подключаем его к шине на макетной плате в соответвтвии с полярностью. В результате должно получиться примерно такое подключение:



Рассмотрим примеры скетчей для каждого из датчиков, используемых в данной схеме:

Датчик Газа

```
int sensorPin = 3;
int gas = 0;
void setup()
{
Serial.begin(9600);
}
void loop()
{
gas = analogRead(sensorPin);
Serial.println(gas);
delay(1000);
}
```

Барометр / Термометр

```
#include <Wire.h>
#include <troyka-imu.h>
Barometer barometer;
void setup()
{
  Serial.begin(9600);
  Serial.println("Begin init...");
  barometer.begin();
  Serial.println("Init completed");
}
void loop()
{
  float pressure = barometer.readPressureMillibars();
  float altitude = barometer.pressureToAltitudeMeters(pressure);
```

```
  float temperature = barometer.readTemperatureC();
  Serial.print("p: ");
  Serial.print(pressure);
  Serial.print(" mbar \t");
  Serial.print("h: ");
  Serial.print(altitude);
  Serial.print(" m \t");
  Serial.print("t: ");
  Serial.print(temperature);
  Serial.println(" C");
  delay(1000);
}
```

Датчик огня

```
int flamePin = 10;
void setup()
{
  pinMode( flamePin, INPUT );
  }
void loop()
{
 {
  Serial.begin(9600);
  }
  if ( !digitalRead( flamePin ) )
    else
  {
    Serial.println ("ОБНАРУЖЕНО ВОЗГОРАНИЕ !!!!");
  delay(1000);
  }
}
```

В данном случае, результатом работы каждого из скетчей является вывод данных в COM-порт и соответствено на экран компьютера через терминальную программу (в нашем случае Putty). При работе на макетном стенде, как и в жизни, нам этого не достаточно. Кроме простого снятия показаний и их отображения на экране, должны выполняться какие-то действия на основе этих показаний. Например, срабатывание световой или звуковой сигнализации, отправка СМС о событии на номер владельца или в МЧС, включение/выключение различных приборов и тому подобное.

Так как напряжение питания электроприборов выше, чем у контроллера, нам потребуются дополнительные устройства – реле для управления включением и отключением питания мощной нагрузки. Мы будем использовать реле с напряжением питания катушек 12В. На одной плате находятся сразу 4 реле, объединенные в один блок:

Управляющий сигнал на реле приходит с платы контроллера. В случае с Arduino, удобней использовать цифровые выводы платы, установив для них в скетче режим с помощью PinMode – «OUTPUT».

Теперь для датчика огня скетч будет выглядеть вот так:

```
int flamePin = 10;
int LED_Pin = 13;
void setup()
{
 Serial.begin(9600);
 pinMode( flamePin, INPUT );
 pinMode( LED_Pin, OUTPUT );
}
void loop()
{
 if ( digitalRead( flamePin ) = = 1)
 digitalWrite( LED_Pin, LOW );
 else
 {
 digitalWrite( LED_Pin, HIGH );
 Serial.println ("ОБНАРУЖЕНО ВОЗГОРАНИЕ !!!!");
 delay(1000);
 }
}
```

То есть, в нашей программе появилочь исполнительное устройство (например, линейка светодиодов), подключенное через реле, которое срабатывает при команде с контроллера, если сработал датчик огня.

Теперь объединим программы для нескольких датчиков в один скетч, что бы смоделировать ситуацию «Пожар» полностью. При срабатывании 3-х датчиков одновременно, мы будем подавать сигнал на реле, которое будет включать световое оповещение.

```
#include <Wire.h>
#include <troyka-imu.h>
Barometer barometer;
int flamePin = 10;
int LED_Pin = 13;
int sensorPin = 3;
int gas = 0;
int temperature = 0;
void setup()
{
 pinMode( flamePin, INPUT );
 pinMode( LED_Pin, OUTPUT );
 Serial.begin(9600);
 Serial.println("Begin init...");
 barometer.begin();
 Serial.println("Init completed");
}
void loop()
{
 if ( digitalRead( flamePin ) == 1)
 {
 digitalWrite( LED_Pin, LOW );
 }
 float temperature = barometer.readTemperatureC();
  if( temperature < 35)
 {
  Serial.print(temperature);
```

```
 Serial.println(" C");
}
if ( analogRead(sensorPin) < 145)
{
 gas = analogRead(sensorPin);
 Serial.println(gas);
}
else
{
digitalWrite( LED_Pin, HIGH );
Serial.println ("ОБНАРУЖЕНО ВОЗГОРАНИЕ !!!!");
delay(1000);
}
}
```

Мы рассмотрели особености работы отдельных групп датчиков и принципы объединения прорамм для них в один скетч.
В процессе изучения предложенных в начале брошюры ситуаций, необходимо учитывать этот опыт для создания одной законченной системы с различным функционалом, объединенным в целостное решение.