Горбатенков Антон, 9 класс, ГАОУ “Долгопрудненская гимназия”, г. Долгопрудный

Создание системы предотвращения протечек и перекачки жидкостей на контроллере Arduino

Каждый день можно встретить различны бытовые проблемы, которые решаются с помощью электроники. Рассмотрим, например, систему обратного осмоса. Это один лучших способов фильтрации воды, но к сожалению, крайне неэкономичный. На один литр отфильтрованной воды приходится пять литров непригодной для питья. И вместо того, чтобы вылить всю эту воду в канализацию, куда выгоднее перекачать воду и, например, поливать ей огород. Или же, наверное, каждый человек хоть раз задумывался сколько проблем будет если прорвет водопровод. Чтобы обезопасить себя от подобных проблем, можно создать систему, которая при обнаружении протечки сразу же перекрывает сантехнический узел. И проблем, которые решаются с помощью робототехники гораздо больше. Я в этом проекте попытаюсь реализовать две вышеописанных системы.

Задачи работы: с минимальными затратами создать на контроллере Arduino прототип системы предотвращения протечек и перекачки жидкостей.

Оборудование: Контроллер Arduino Uno (Китай), шаговый двигатель от неисправного принтера, насос, модули управления нагрузкой XY-MOS (Китай), блок питания, мультиметр, компьютер, макетная плата, провода, электронные компоненты (транзистор, резисторы, диоды, светодиоды, кнопка), стеклотекстолит.

Часть 1. Автоматическая система перекачки жидкостей.

Основная идея реализации этой части системы – это три электрода на разных уровнях, помещенные в емкость с накапливающейся жидкостью. На самый нижний из них подается напряжение, со среднего и с верхнего напряжение считывается. Если резервуар заполняется водой, напряжение с нижнего электрода поступает на верхний, что является сигналом для запуска насоса, откачивающего воду. Для чего нужен средний электрод? Выключать насос сразу после того, как с верхнего электрода пропадает напряжение - бессмысленно. Так как верхний электрод находится на верхнем уровне резервуара, то сразу же, как уровень воды станет немного ниже максимального, перекачка воды закончится. И тогда резервуар будет почти всегда полный. Поэтому насос выключается, когда пропадает напряжение на среднем электроде, который находится близко к нижнему, то есть у дна резервуара.

Для реализации всей системы мною был выбран самый распространенный контроллер – Arduino Uno. Прежде я не имел серьезного опыта работы с ним, так что для проекта мне пришлось многое изучать.

Аппаратная часть системы довольно проста. Датчик определения уровня жидкости выполнен из фольгированного стеклотекстолита в виде трех электродов, работающих по описанной выше схеме. В процессе изготовления потребовалось вытравливание подготовленной платы в хлорном железе, где слой меди, за исключением электродов (контактов) - растворился.

Так как нагрузочная способность выходов ардуино - низкая, а насос – довольно мощная нагрузка, то понадобилось включать его от дополнительного источника питания через транзисторный ключ, управляемый выходом контроллера.

Главной проблемой при реализации прототипа было то, что определение напряжения на среднем и верхнем электродах было нестабильным, и определялось иногда как логическая единица, а иногда как логический ноль. Проблема была решена «подтяжкой» электродов к земле резисторами.

В коде программы алгоритм работы подсистемы описан в виде отдельного блока (с подробными комментариями) в составе общего для работы всей системы цикла обработки. Текст программы прилагается отдельным файлом (текстовый файл с расширением. ino, открывается в любом текстовом редакторе, например, notepad++).

Схема соединений приведена ниже.

Часть 2. Система предотвращения протечек

Датчик протечек также выполнен в виде подсистемы на том же контроллере, так как ресурсов контроллера достаточно для реализации обеих частей.

Его идея заключается в том, что в месте потенциальной протечки располагается датчик, представляющий из себя два электрода, на одном из которых постоянно присутствует напряжение логического нуля. Второй электрод «подтянут» внутренним резистором контроллера к напряжению питания и с него в цикле считывается значение. Если протекает вода, то напряжение на втором электроде падает до уровня первого.

Контроллер определяет это изменение и подает сигнал на шаговый двигатель, который, вращая водопроводный кран, перекрывает водоснабжение.

Также в системе предусмотрена индикация на светодиодах. Красный светодиод сообщает о протечке и затухает, когда протечка устранена. Желтый - сообщает о том, что кран находится в закрытом положении после вращения двигателя. Если он не горит, то кран – открыт.

Так как факт окончательного устранения причин протечки может определить только человек, то в системе предусмотрена кнопка для ручного управления краном. Если датчик сообщает об отсутствии протечки, то по нажатию кнопки закрытый кран - открывается.

По кнопке также может быть включена и дополнительная функция – профилактический поворот крана на 360 градусов. Функция может быть включена только при условии открытого крана и отсутствия протечек. Такая профилактика необходима потому, что в условиях длительного бездействия при жесткой воде – кран может частично потерять свои возможности по вращению из-за отложения солей на его внутренних поверхностях. Периодический же поворот крана устраняет эту проблему.

В качестве шагового двигателя я использовал шаговый двигатель от старого неисправного принтера. Для управления двигателем на его обмотки должна подаваться определенна последовательность импульсов. Эта последовательность формируется программой. Так как двигатель – мощная нагрузка, то также используется внешний источник питания, а управляющие сигналы от контроллера подаются через модули управления мощной нагрузкой на полевых транзисторах.

Основная проблема заключалась в том, что обе системы выполнены на одном контроллере. Но контроллер не может по умолчанию выполнять задачи параллельно, поэтому задержка в одной подсистеме вносит также и задержку в работу другой. Например, в первой системе планировалось опрашивать датчик с периодичностью в 10 секунд. Но задержка в 10 секунд могла бы стать критичной в случае протечки. В целом, проблема оказалась надуманной, так как сокращение периодичности опроса датчика уровня – никак не влияет на работу системы. Поэтому я принял решение поставить небольшую задержку в 1 секунду, позволяющую надежно получать урони датчика и не критичную для протечки.

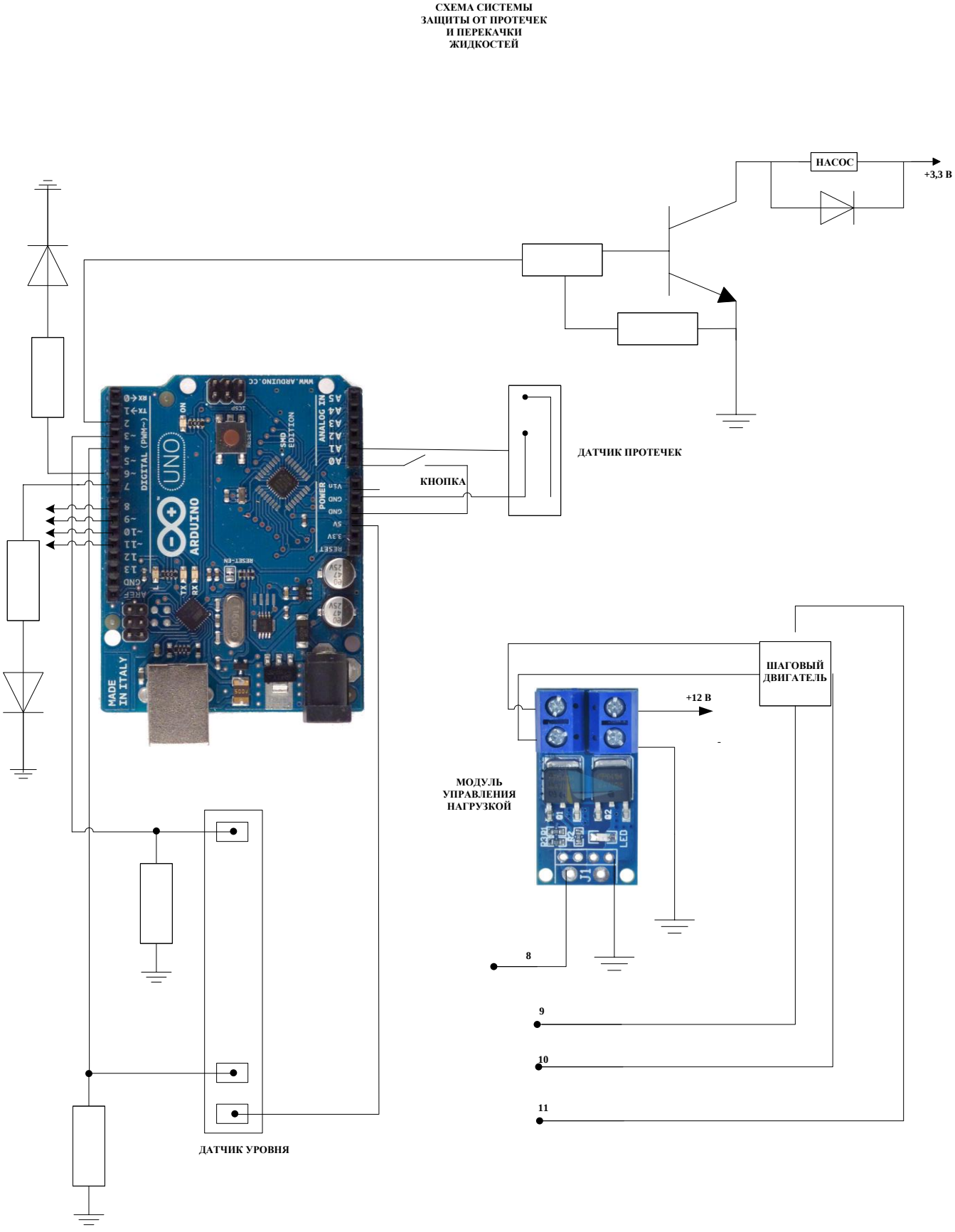
Вывод:

На момент выполнения работы, мне удалось создать рабочий прототип системы. Обе подсистемы работают как задумано изначально.

Система пригодна и для использования на практике. Однако, есть множество идей для ее оптимизации. Например, связь контроллера с датчиками протечки и уровня можно реализовать на базе радио-модулей или Wi-Fi, а не проводами. Также возможно реализовать систему на базе простого микроконтроллера, а не промышленной платы Arduino, что уменьшит и его размеры, и стоимость.

В процессе работы я узнал для себя много нового в сфере электроники и робототехники, научился самостоятельно паять, изготавливать печатные платы, собирать электронные схемы, программировать микроконтроллеры. Я нахожу это крайне интересным занятием и надеюсь в будущем дальше развиваться в этой области.

Электрическая схема



Программа микроконтроллера:

```
#include <avr/eeprom.h>            //библиотека работы с EEPROM МК

/*номера выводов МК*/
  /*датчик уровня жидкости*/
  const byte RUN_PUMP = 2;         // включение/выключение насоса
  const byte MAX_PUMP_LEVEL = 3;   // максимальный уровень датчика
  const byte STOP_PUMP_LEVEL = 4;  // уровень остановки насоса датчика (средний)

  /*датчик протечки*/
  const byte LEAK_IN = A1;         // вход сигнала протечки

  /*управление шаговым двигателем водопроводного крана*/
  const byte TAP_COIL_1 = 8;       // обмотки 1-4
  const byte TAP_COIL_2 = 9;
  const byte TAP_COIL_3 = 10;
  const byte TAP_COIL_4 = 11;

  /*индикация, управление*/
  const byte LEAK_LED = 6;          // сигнализатор протечки
  const byte CLOSED_LED = 7;        // сигнализатор закрытого крана
  const byte ROTATE_BUTTON = A0;    // кнопка ручного управления краном
/*номера выводов МК*/


/*настройки программы*/
  const float STEP_ANGLE = 7.5;     // угол поворота за шаг двигателя крана
  const int CLOSE_ANGLE = 90;       // угол поворота для закрывания крана
  const int OPEN_ANGLE = 270;       // угол поворота для открывания закрытого крана
  const int ROTATE_ANGLE = CLOSE_ANGLE + OPEN_ANGLE;  // угол полного оборота
  const int STEP_LENGTH = 25;       // длительность формируемого импульса шага двигателя
  const int CHECK_DELAY = 1000;     // задержка между циклами проверки


/*глобальные переменные*/
  bool isLeaked = false;            // флажок обнаружения протечки
  bool isClosed = false;            // флажок закрытого крана
  bool isPumping = false;           // флажок включенной перекачки


/*процедура инициализации контроллера*/
void setup() {
  pinMode(RUN_PUMP, OUTPUT);        // выходы МК (OUTPUT)
  pinMode(MAX_PUMP_LEVEL, INPUT);   // входы МК (INPUT)
  pinMode(STOP_PUMP_LEVEL, INPUT);
  pinMode(LEAK_LED, OUTPUT);
  pinMode(LEAK_IN, INPUT_PULLUP);   // входы МК с внутренней подтяжкой к высокому уровню(INPUT_PULLUP)
  pinMode(ROTATE_BUTTON, INPUT_PULLUP);
  pinMode(CLOSED_LED, OUTPUT);
  pinMode(TAP_COIL_1, OUTPUT);
  pinMode(TAP_COIL_2, OUTPUT);
  pinMode(TAP_COIL_3, OUTPUT);
  pinMode(TAP_COIL_4, OUTPUT);

  // чтение флажка закрытого крана из EEPROM (на случай аварийного завершения предыдущего сеанса работы)
  isClosed = eeprom_read_byte(0);

  // инициализация обмена с контроллером в режиме отладки
  //Serial.begin(9600);
}
```

```
/*процедура цикла выполнения*/
void loop()
{
  /* автоматическое управление краном при протечке */
  bool leak = digitalRead(LEAK_IN);                       // чтение датчика протечки

  if (!isLeaked && leak == LOW)                           // обнаружена протечка, но флажок еще не установлен
  {
    isLeaked = ChangeFlag(isLeaked, LEAK_LED);            // установить флажок протечки
    if (!isClosed)                                        // если кран не закрыт - закрыть
      RotateWaterTap(CLOSE_ANGLE);
  }

  if (isLeaked && leak == HIGH)                           // протечка устранена, но флажок протечки еще не снят
     isLeaked = ChangeFlag(isLeaked, LEAK_LED);           // сбросить флажок протечки
  /* автоматическое управление краном при протечке */


  /* ручное управление краном */
  bool buttonPressed = digitalRead(ROTATE_BUTTON) == LOW; // чтение кнопки
  if (!isLeaked && buttonPressed)                         // если протечки нет
    RotateWaterTap(isClosed ? OPEN_ANGLE : ROTATE_ANGLE); // если кран закрыт - открыть,
                                                          // если открыт - провернуть полностью (режим предотвращение "закисания")
  /* ручное управление краном */


  /* автоматическое управление насосом */
  if (isPumping && digitalRead(STOP_PUMP_LEVEL) == LOW)   // если при работающей перекачке жидкость достигла среднего уровня датчика
     isPumping = ChangeFlag(isPumping, RUN_PUMP);         // выключить насос

  if (!isPumping && digitalRead(MAX_PUMP_LEVEL) == HIGH)  // если жидкость достигла максимального уровня датчика
    isPumping = ChangeFlag(isPumping, RUN_PUMP);          // включить насос

  delay (CHECK_DELAY);
}
```

```
  if (!isPumping && digitalRead(MAX_PUMP_LEVEL) == HIGH)  // если жидкость достигла максимального уровня датчика
    isPumping = ChangeFlag(isPumping, RUN_PUMP);          // включить насос

  delay (CHECK_DELAY);
}


/* функция установки/сброса флажка        */
/* flag - флажок                          */
/* pin - вывод МК для выполнения действия */
/* return flag - новое значение флажка    */
bool ChangeFlag(bool flag, byte pin)
{
  flag = !flag;
  digitalWrite(pin, flag);
  return flag;
}


/* процедура поворота крана (формирования последовательности импульсов )     */
/* формирует последовательность импульсов для управления шаговым двигателем */
/* initAngle - угол поворота                                                 */
void RotateWaterTap(int initAngle)
{
  float currentAngle = 0;
  byte stepNumber = 0;

  while (currentAngle < initAngle)                      // пока поворот выполнен не на заданный угол
  {
    DoStep(stepNumber);                                 // повернуть на один шаг
    currentAngle += stepNumber == 0 ? 0 : STEP_ANGLE;   // вычислить значение  текущего угла поворота
    stepNumber++;                                       // инкремент номера шага поворота
    if (stepNumber > 4)
      stepNumber = 1;
  }
  DoStep(0);                                            // сбросить последовательность для поворота
  if (initAngle < ROTATE_ANGLE)                         // если поворот крана не профилактический
  {
    isClosed = ChangeFlag(isClosed, CLOSED_LED);        // то установить/сбросить флажок закрытого крана
    eeprom_write_byte(0, isClosed);                     // и сохранить его в EEPROM на случай аварии
  }
}

/* процедура выполнения шага двигателя                  */
/* формирует импульс на выводах МК для выполнения шага  */
/* stepNumber - номер шага в последовательности         */
void DoStep(int stepNumber)
{
  switch (stepNumber)                     // в зависимости от номера
  {
    case 1:
    {
      digitalWrite(TAP_COIL_3, LOW);      // импульс формируется на
      digitalWrite(TAP_COIL_1, HIGH);     // соответствующей номеру шага обмотке двигателя
      break;
    }
    case 2:
    {
      digitalWrite(TAP_COIL_4, LOW);
      digitalWrite(TAP_COIL_2, HIGH);
      break;
    }
    case 3:
    {
      digitalWrite(TAP_COIL_1, LOW);
      digitalWrite(TAP_COIL_3, HIGH);
      break;
    }
    case 4:
    {
      digitalWrite(TAP_COIL_2, LOW);
      digitalWrite(TAP_COIL_4, HIGH);
      break;
    }
    default:
    {
      digitalWrite(TAP_COIL_1, LOW);      // сброс на всех выводах для шага не из последовательнолсти
      digitalWrite(TAP_COIL_2, LOW);
      digitalWrite(TAP_COIL_3, LOW);
      digitalWrite(TAP_COIL_4, LOW);
    }
  }
  if (stepNumber > 0)                     // длительность формируемого импульса
    delay(STEP_LENGTH);
}
```

Текст программы прилагается отдельным файлом (текстовый файл с расширением. ino, открывается в любом текстовом редакторе, например, notepad++).