Государственное автономное профессиональное образовательное учреждение
Свердловской области
«ЕКАТЕРИНБУРГСКИЙ ТЕХНИКУМ «АВТОМАТИКА»

Исследовательский проект

«Система измерения температуры с использование контроллера Arduino»

Руководитель

Лунегов О.Б., преподаватель профессионального цикла

Выполнил

Волегов С.О., обучающийся группы ПКС-31

2018 г.

Оглавление

Система автоматического измерения и контроля температуры в жилых помещениях

Введение

Измерение температуры с использованием Arduino

Актуальность

Система автоматического измерения температуры в помещении довольно полезная вещь и может быть использована во многих отраслях. В нашем случае эта система будет применена для решения таких проблем как:

- Пожарная безопасность
- Контроль температуры в помещении
- Контроль за системой отопления

Рассмотрим каждую проблему детально.

Пожарная безопасность

Наш датчик температуры позволят замерять температуру окружающей среды с минимальной погрешностью. Установим предел температуры до такого показателя, до которого система отопления не сможет нагреть комнату, но до которого её бы смог нагреть, возникший в квартире пожар. Теперь, когда температура поднимется выше этого предела контроллер с подключённой к нему платой расширения подаст звуковой сигнал и выдаст на цифровой дисплей номер датчика, который зафиксировал повышение температуры. Главным преимуществом такой системы перед датчиками дыма, которые используются для тех же целей оповещения о пожаре, является невозможность блокирования такого датчика посторонними предметами, т.к. реагирует он не на дым, а на температуру воздуха. Помимо вывода на цифровой дисплей контроллер также может быть подключен к компьютеру, что позволит считывать показания с его экрана. Также подключении

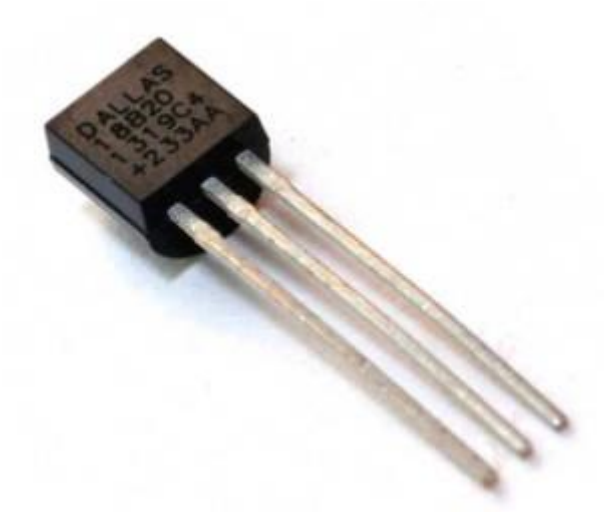
дополнительных модулей к контроллеру, будет возможен вывод показаний на сайт в сеть интернет, что позволит контролировать температуру даже на расстоянии.

Контроль температуры в помещении

Ещё одна немаловажная функция которая может быть реализована с помощью нашей системы, это функция контроля температуры.

Устройство системы измерения температуры

Датчик температуры DS18B20



В качестве измерителя температуры мы будем использовать датчик – DS18B20, использующий интерфейс 1-Wire для связи с Arduino

Техническое описание датчика:

DS18B20 - цифровой измеритель температуры, с разрешением преобразования 9 - 12 разрядов и функцией тревожного сигнала контроля за температурой. Параметры контроля могут быть заданы пользователем и сохранены в энергонезависимой памяти датчика.

DS18B20 обменивается данными с микроконтроллером по однопроводной линии связи, используя протокол интерфейса 1-Wire.

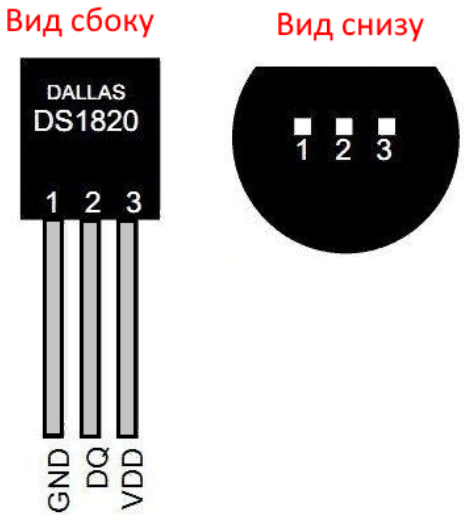
Питание датчик может получать непосредственно от линии данных, без использования внешнего источника. В этом режиме питание датчика происходит от энергии, запасенной на паразитной емкости.

Диапазон измерения температуры составляет от -55 до +125 °C. Для диапазона от -10 до +85 °C погрешность не превышает 0,5 °C.

Характеристики датчика:

- Для однопроводного интерфейса 1-Wire достаточно одного порта связи с контроллером.
- Каждое устройство имеет уникальный серийный код длиной 64 разряда.
- Возможность подключения нескольких датчиков через одну линию связи.
- Нет необходимости во внешних компонентах.
- Возможность получать питание непосредственно от линии связи. Напряжение питания в пределах 3,0 В … 5,5 В.
- Диапазон измерения температуры -55 ... +125 °C.
- Погрешность не превышает 0,5 °C в диапазоне -10 ... +85 °C.
- Разрешение преобразования 9 … 12 бит. Задается пользователем.
- Время измерения, не превышает 750 мс, при максимально возможном разрешении 12 бит.
- Возможность программирования параметров тревожного сигнала.
- Тревожный сигнал передает данные об адресе датчика, у которого температуры вышла за заданные пределы.
- Совместимость программного обеспечения с DS1822.

Выводы датчика:



8-PIN SOIC	TO-92	СИГНАЛ	ОПИСАНИЕ
5	1	GND	Земля
4	2	DQ	Вывод сигнала данных (входа/выход). Выход типа открытый коллектор интерфейса 1-Wire. Также через него происходит питание в режиме ”паразитное питание”.
3	3	V_{DD}	Вывод внешнего питания. В режиме ”паразитного питания" должен быть подключен к земле.

Режим – измерение температуры.

Основная функция DS18B20 – преобразование температуры датчика в цифровой код. Разрешение преобразования задается 9, 10, 11 или 12 бит. Это соответствует разрешающей способность - 0,5 (1/2) °C, 0,25 (1/4) °C, 0,125 (1/8) °C и 0,0625 (1/16) °C. При включении питания, состояние регистра конфигурации устанавливается на разрешение 12 бит.

После включения питания DS18B20 находится в низко-потребляющем состоянии покоя. Чтобы инициировать измерение температуры мастер (микроконтроллер) должен выполнить команду ПРЕОБРАЗОВАНИЯ ТЕМПЕРАТУРЫ [44h]. После завершения преобразования, результат

измерения температуры будет находиться в 2 байтах регистра температуры, и датчик опять перейдет в состояние покоя.

Если DS18B20 включен по схеме с внешним питанием, то мастер может контролировать состояние команды конвертации. Для этого он должен читать состояние линии (выполнять временной слот чтения), по завершению команды, линия перейдет в высокое состояние. Во время выполнения команды конвертации линия удерживается в низком состоянии.

При питании от заряда паразитной емкости, такой способ не допустим, т.к. во время операции преобразования на шине необходимо удерживать высокий уровень сигнала для питания датчика. Технология “паразитного питания” подробно описывается в разделе ПИТАНИЕ DS18B20.

DS18B20 измеряет температуру в градусах по шкале Цельсия. Результат измерения представляется как 16-разрядное, знаковое число в дополнительном коде (рис. 2.) . Бит знака (S) равен 0 для положительных чисел и равен 1 для отрицательных. При разрешении 12 бит, у регистра температуры все биты значащие, т.е. имеют достоверные значения. Для разрешения 11 бит, не определен бит 0. Для 10-битного разрешения не определены биты 0, 1. При разрешении 9 бит, не достоверное значение имеют биты 0, 1 и 2. В таблице 2 показаны примеры соответствия цифровых кодов значению температуры.

Рисунок 2. Формат регистра температуры

	бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
Мл. байт	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	бит 15	бит 14	бит 13	бит 12	бит 11	бит 10	бит 9	бит 8
Ст. байт	S	S	S	S	S	2^6	2^5	2^4

Таблица 2. Соответствие данных и температуры.

ТЕМПЕРАТУРА	ЦИФРОВОЙ КОД (двоичный)	ЦИФРОВОЙ КОД (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C	0000 0101 0101 0000	0550h
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FE6Fh
-55°C	1111 1100 1001 0000	FC90h

Для людей не искушенных в двоичной математике, напишу, что для вычисления температуры надо:

При положительном значении (S=0) код перевести в десятичный и умножить на 0,0625 °C.

При отрицательном значении (S=1) сначала необходимо перевести дополнительный код в прямой. Для этого надо инвертировать каждый разряд двоичного кода и прибавить 1. А затем перевести в десятичный и умножить на 0,0625 °C.

Режим – передача тревожного сигнала.

После выполнения команды преобразования температуры, измеренное значение сравнивается с верхним и нижним порогами из регистров Th, Tl (формат на рисунке 3). Это байтовые значения, знаковые, в дополнительном

коде, S =0 означает, что число положительное, а S=1 – отрицательное. Хранятся пороговые значения в энергонезависимой памяти (EEPROM). Th и Tl доступны для чтения и записи через байты 2, 3 оперативной памяти. Подробно об этом в разделе ПАМЯТЬ.

Рисунок 3. Формат регистров TH и TL

бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	бит 1	бит 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Из-за разной длины регистров TH, TL и температуры, они сравниваются только с битами 11 по 4 регистра температуры. Если значение измеренной температуры превышает TH или ниже, чем TL, то формируется признак аварии в DS18B20. Признак перезаписывается с каждым измерением температуры, и если температура возвращается в заданные пределы, то он сбрасывается.

Ведущее устройство может проверить состояние признака аварии с помощью команды ПОИСК ТРЕВОЖНОГО СИГНАЛА [ECh]. Любой датчик с активным признаком ответит на команду поиска. Таким образом, мастер точно определит, какой DS18B20 вырабатывает сигнал тревоги. После изменения значений регистров TH и TL, только следующее преобразование температуры сформирует достоверный признак тревоги.

Arduino-контроллер

Для обработки сигналов, поступающих с датчика мы будем использовать контроллер Arduino Leonardo

Техническое описание Arduino Leonardo:

Arduino Leonardo — контроллер на базе микроконтроллера **ATmega32u4.**

Платформа имеет 20 цифровых вход/выходов (7 из которых могут использоваться как выходы ШИМ и 12 как аналоговые входы), кварцевый генератор 16 МГц, разъем микро-USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи.

В отличие от всех предыдущих плат **ATmega32u4** имеет встроенную поддержку для USB соединения, это позволяет задать как **Leonardo** будет виден при подключении к компьютеру, это может быть клавиатура, мышь, виртуальный серийный / COM порт.

Характеристики Arduino Leonardo:

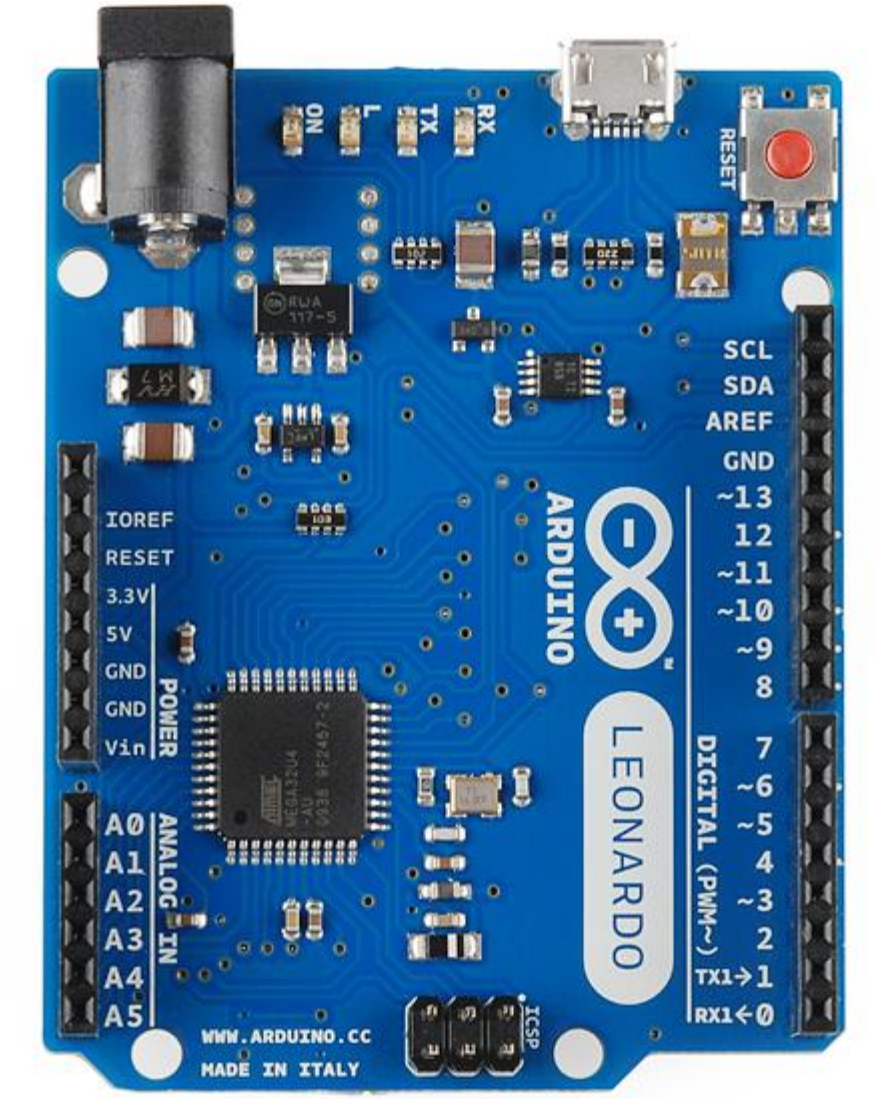
Микроконтроллер	ATmega32u4
Рабочее напряжение	5 В
Входное напряжение (рекомендуемое)	7-12 В
Входное напряжение (предельное)	6-20 В
Цифровые Входы/Выходы	20 (7 из которых могут использоваться как выходы ШИМ)
Аналоговые каналы	12
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 Кб (ATmega32u4) из которых 4 Кб используются для загрузчика
ОЗУ	2 Кб (ATmega32u4)
EEPROM	1 Кб (ATmega32u4)
Тактовая частота	16 МГц

Питание

Arduino Leonardo может получать питание через подключение USB или от внешнего источника питания. Источник питания выбирается автоматически.

Внешнее питание (не USB) может подаваться через преобразователь напряжения AC/DC (блок питания) или аккумуляторной батареей. Преобразователь напряжения подключается посредством разъема 2.1 мм с центральным положительным полюсом. Провода от батареи подключаются к выводам Gnd и Vin разъема питания.

Платформа может работать при внешнем питании от 6 В до 20 В. При напряжении питания ниже 7 В, вывод 5V может выдавать менее 5 В, при этом платформа может работать нестабильно. При использовании напряжения выше 12 В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон от 7 В до 12 В.



Выводы питания:

- **VIN**. Вход используется для подачи питания от внешнего источника (в отсутствие 5 В от разъема USB или другого регулируемого источника питания). Подача напряжения питания происходит через данный вывод.

- **5V**. Регулируемый источник напряжения, используемый для питания микроконтроллера и компонентов на плате. Питание может подаваться от вывода VIN через регулятор напряжения, или от разъема USB, или другого регулируемого источника напряжения 5 В.
- **3V3**. Напряжение на выводе 3.3 В генерируемое встроенным регулятором на плате. Максимальное потребление тока 50 мА.
- **GND**. Выводы заземления.
- **IOREF**. Вывод с рабочим напряжением вход/выходов платы.
  Для **Leonardo** это 5 В. Предполагается к использованию платами расширения для правильноего выбора рабочего напряжения.

Память

Микроконтроллер ATmega32u4 располагает:

- 32 кБ флэш памяти, из которых
- 4 кБ используется для хранения загрузчика, а также 2.5 кБ ОЗУ (SRAM)
- 1 Кб EEPROM.(которая читается и записывается с помощью библиотеки EEPROM).

Входы и Выходы

Каждый из 20 цифровых выводов Leonardo может настроен как вход или выход, используя функции pinMode(), digitalWrite(), и digitalRead(), . Выводы работают при напряжении 5 В. Каждый вывод имеет нагрузочный резистор (по умолчанию отключен) 20-50 кОм и может пропускать до 40 мА.
Некоторые выводы имеют особые функции:

- **Последовательная шина: 0 (RX) и 1 (TX)**. Выводы используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы последовательной шины ATmega32U4 USB-to-TTL. Обратите внимание что у **Leonardo**, класс **Serial** относится к последовательному соединению USB CDC. Последовательное соединение через выводы 0 и 1 осуществляется через класс **Serial1.**
- **TWI: 2 (SDA) и 3 (SCL)**. Посредством выводов осуществляется связь I2C (TWI), для создания которой используется библиотека Wire.
- **Внешнее прерывание: 2 и 3**. Данные выводы могут быть сконфигурированы на вызов прерывания либо на младшем значении, либо на переднем или заднем фронте, или при изменении значения. Подробная информация находится в описании функции attachInterrupt().
- **ШИМ: 3, 5, 6, 9, 10, 11 и 13.** Любой из выводов обеспечивает ШИМ с разрешением 8 бит при помощи функции analogWrite().
- **SPI: на разъеме ICSP.** Посредством данных выводов осуществляется связь SPI, для чего используется библиотека SPI. Обратите внимание, что в Leonardo выводы SPI не разведены на цифровые вход/выходы как это было в предыдущих версиях Arduino контроллеров.
- **LED: 13.** Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий потенциал, то светодиод горит.
- Аналоговые входы: A0-A5, A6-A11 (на цифровых выводах 4, 6, 8, 9, 10 и 12). Leonardo имеет 12 аналоговых входов, помеченных от A0 до A11. Все анадоговые входы могут работать в режиме цифровых вход/выходов. Входы с A0 по A5 совпадают с аналоговыми входами UNO. Входы с A6 по A11 на цифровых выводах 4, 6, 8, 9, 10 и 12 соответственно. Рарешение аналоговых входов — 10 бит, т.е. 1024 различных значения. По умолчанию значение на аналоговых входах измеряется от земли (0) до 5 Вольт, верхний предел диапазона может быть изменен с помощью AREF входа и analogReference() функции.

Дополнительная пара выводов платформы:

- **AREF.** Опорное напряжение для аналоговых входов. Используется с функцией analogReference().
- **Reset.** Низкий уровень сигнала на выводе перезагружает микроконтроллер. Обычно применяется для подключения кнопки перезагрузки на плате расширения, закрывающей доступ к кнопке на самой плате Arduino.

Обратите внимание на соединение между выводами Arduino и портами ATmega328.

Связь

На платформе **Arduino Leonardo** может устанавливать связь с компьютером, другими устройствами Arduino или микроконтроллерами несколькими способами. ATmega32U4 поддерживают последовательный интерфейс UART TTL (5 В), осуществляемый выводами 0 (RX) и 1 (TX). ATmega32U4 позволяет также организовать последовательное соединение с программами на стороне компьютера через USB так, чтобы они "общались" с платой через виртуальный COM порт. Leonardo с помощью стандартных драйверов USB COM (для Windows потребуется .inf файл) может подключаться как USB 2.0 устройство. Мониторинг последовательной шины (Serial Monitor) среды разработки Arduino позволяет посылать и получать текстовые данные при подключении к платформе. Светодиоды RX и TX на платформе будут мигать при передаче данных через USB подключение (но не при использовании последовательной передачи через выводы 0 и 1).

Библиотекой SoftwareSerial возможно создать последовательную передачу данных через любой из цифровых выводов Leonardo.

ATmega32U4 поддерживает интерфейсы I2C (TWI) и SPI. В Arduino включена библиотека Wire для удобства использования шины I2C. Для SPI может использоваться библиотека SPI.

Arduino Leonardo умеет определяться при подключении к компьютеру как устройство мышь или клавиатура. Управление этим режимом осуществляется через классы Keyboard и Mouse.

Программирование

Платформа программируется посредством ПО Arduino. Из меню **Tools > Board** выбирается «Arduino Leonardo» (согласно установленному микроконтроллеру). Подробная информация находится в справочнике и инструкциях.

Микроконтроллер ATmega32U4 на Leonardo поставляется с записанным загрузчиком, опрощающим запись новых программ без использования внешних программаторов. Связь осуществляется протоколом AVR109.

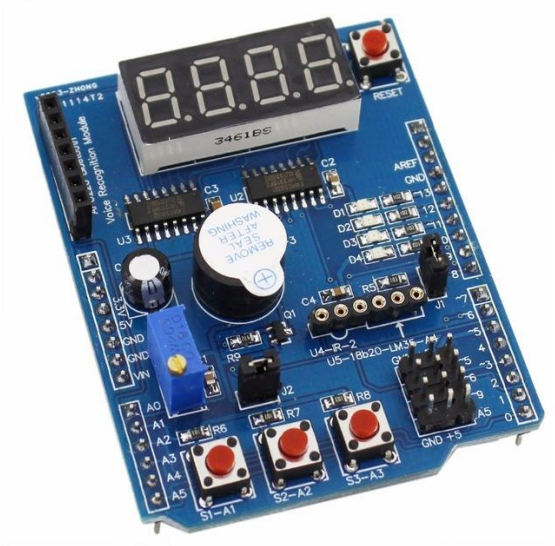
Имеется возможность не использовать загрузчик и запрограммировать микроконтроллер через выводы ICSP (внутрисхемное программирование). Подробная информация находится в данной инструкции.

Автоматическая (программная) перезагрузка

Leonardo разработана таким образом, чтобы перед записью нового кода перезагрузка осуществлялась самой программой Arduino на компьютере, а не нажатием кнопки на платформе. Перезагрузка срабатывает когда виртуальный CDC COM порт открывается со скоростью 1200 бод, а затем закрывается. Когда это происходит, микропроцессор уходит на перезагрузку, разрывая USB соединение. После перезагрузки стартует загрузчик (бутлодер)

и остается активным примерно 8 секунд. Загрузчик также можно инициировать нажатием кнопки Reset. Обратите внимание, что при подачи питания контроллер сразу переходит к выполнению загруженной пользовательской программы без выполнения загрузчика.

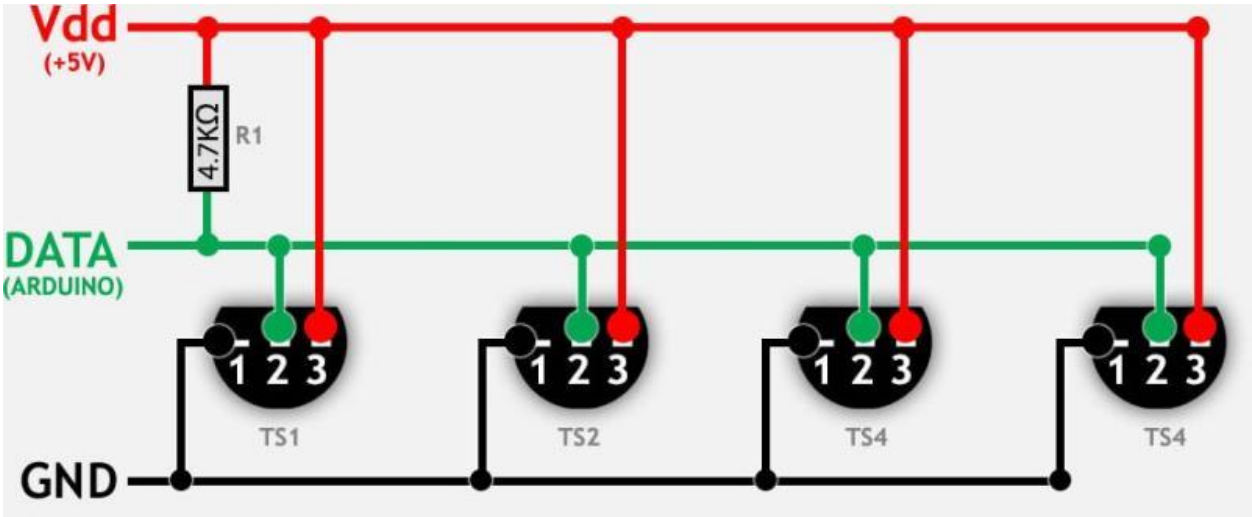
Плата расширения MultiFunctionShield



Помимо контроллера и датчика нам также понадобится устройство для вывода показаний температуры, для этого мы воспользуемся платой расширения MultiFunctionShield. MultiFunctionShield имеет встроенный цифровой интерфейс – что позволит выводить значение температуры определенного датчика на его экран. Также плата имеет встроенный зуммер – что позволит подавать сигналы в случае превышения заданного предельного значения температуры или в случае потери связи с одним из датчиков. Помимо этого, на плате имеется три кнопки с помощью которых можно переключаться между датчиками. Ну и последнее – плата имеет уже встроенный специальный вывод, куда можно подключить каскад датчиков.

Система измерения температуры

Система измерения температуры выглядит следующим образом: каскад, параллельно соединённых, датчиков соединен с платой расширения MultiFunctionShield, для создания необходимого падения напряжения на первом датчике выводы Data и Vdd(питание) должны быть соединены резистором на 1,6 кОм.



Плата расширения подключается к Arduino, на которую уже загружена программа.

Принцип работы

Система измерения температуры циклична т.е. выполняется постоянно, состоит она из следующих шагов:

1. Контроллер отправляет всем датчикам из каскада запрос на измерение температуры после чего ожидает 0.1 секунды и оправляет запрос на возврат значения температуры в память контроллера.
2. Контроллер получив значение температуры преобразует его в читаемый для пользователя вид
3. Далее пользователь должен выбрать с какого датчика ему необходимо получить значение температуры, выбор осуществляется при помощи кнопок на плате MultiFunctionShield, если же пользователь не выбрал нужный ему датчик, то данные будут выводиться с первого датчика из каскада. Температура выводится на дисплей в виде десятичного числа с точностью до десятых. Примечательно что дисплей может выводить температуру не только выше нуля, но и ниже, что позволяет использовать её как в помещении, так и на улице.

Программа измерения температуры

```
#include <OneWire.h>

//Константы
#define LATCH_DIO 4
#define CLK_DIO 7
#define DATA_DIO 8

//Объекты
OneWire  ds(A4); // Датчик температуры

// Разряды цифрового дисплея
byte a = 0b11111111; //
byte b = 0b11111111;
byte c = 0b11111111;
byte d = 0b11111111;
```

```
int timer1 = 0, timer2 = 0;
float celsius;

const byte SEGMENT_MAP[] =
{0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0X80,0X90,0b10111111,0b
11111111}; // Шаблоны сегментов цифрового дисплея
const byte SEGMENT_SELECT[] = {0xF1,0xF2,0xF4,0xF8}; // Адреса
разрядов

// Преустановки
void setup() {
  Serial.begin(9600);
  // Установка режимов для цифрового дисплея
  pinMode(LATCH_DIO,OUTPUT);
  pinMode(CLK_DIO,OUTPUT);
  pinMode(DATA_DIO,OUTPUT);
}


void WriteNumberToSegment(byte Segment, byte Value)
{
  digitalWrite(LATCH_DIO,LOW);
  if(Segment==2)shiftOut(DATA_DIO, CLK_DIO, MSBFIRST,
SEGMENT_MAP[Value] - 0b10000000);
  else shiftOut(DATA_DIO, CLK_DIO, MSBFIRST,
SEGMENT_MAP[Value]);
  shiftOut(DATA_DIO, CLK_DIO, MSBFIRST, SEGMENT_SELECT[Segment]
);
  digitalWrite(LATCH_DIO,HIGH);
}

// Функция вывода значения температуры на дисплей
void ShowTemp()
{
  WriteNumberToSegment(0,a);
  WriteNumberToSegment(1,b);
```

```
  WriteNumberToSegment(2,c);
  WriteNumberToSegment(3,d);
}

// Основной цикл программы
void loop() {
  byte i;
  byte present = 0;
  byte type_s;
  byte data[12]; // Данные получаемые с датчика
  byte addr[8]; // Адрес датчика

  timer1++;
  timer2++;

  ShowTemp(); // Первичный вывод температуры на дисплей

  if(timer1>=100)
  {
  timer1 = 0;
  if ( !ds.search(addr)) {
    ds.reset_search();
    return;
  }
  ShowTemp();

  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      type_s = 1;
      break;
    case 0x28:
      type_s = 0;
      break;
    case 0x22:
      type_s = 0;
```

```
    break;
  default:
    return;
}

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);
}


if(timer2 >= 300)
{
  timer2 = 0;


  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);

  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
  }

  int16_t raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3;
    if (data[7] == 0x10) {
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  }
  else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw & ~7;
    else if (cfg == 0x20) raw = raw & ~3;
    else if (cfg == 0x40) raw = raw & ~1;
```

```
    }
    celsius = (float)raw / 16.0; // Приведение значения
температуры в читаемый формат
  }

  int temp = celsius*100;
  if (celsius < 0)
  {
    a = 10;
    temp * -1;
  }

  a = temp/10000;
  b = (temp / 1000) - (a * 10);
  c = (temp / 100) - (a * 100) - (b * 10);
  d = (temp / 10) - (a * 1000) - (b * 100) - (c * 10);

  if(a==0)a = 11;
}
```

Заключение

Дальнейшее развитие этого проекта, предполагает подключение к системе сетевой платы расширения для вывода результатов на сервер, а также увечить число подключаемых датчиков и добавить возможность выборочного вывода температуры на экран, за счет кнопок на плате расширения Multifunction Shield. Поскольку наша система является частью системы «Умный дом», планируется расширить её подключив к ней ультразвуковой дальномер, который будет прикреплён к окнам для установления их открытого или закрытого состояния.