

Лабораторная работа №3

Аналоговый ввод в среде Arduino, аналоговые датчики.

Аналого-цифровой преобразователь

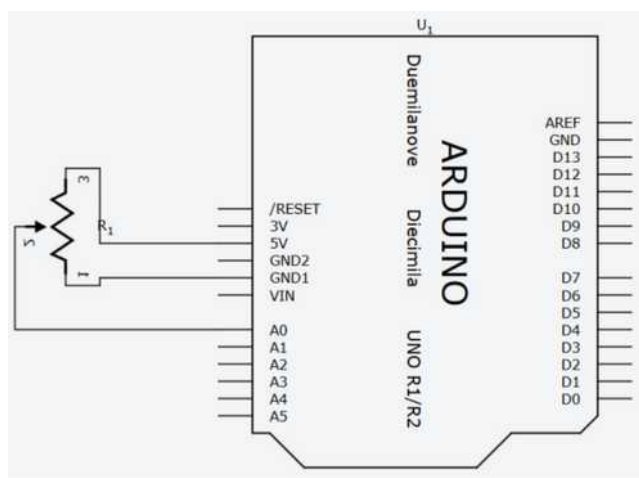
Микроконтроллеры Atmega, используемые в Arduino, содержат шестиканальный аналого-цифровой преобразователь (АЦП). Разрешение преобразователя составляет 10 бит, что позволяет на выходе получать значения от 0 до 1023. Основным применением аналоговых входов большинства платформ Arduino является чтение аналоговых датчиков, но в тоже время они имеют функциональность вводов/выводов широкого применения (GPIO) (то же, что и цифровые порты ввода/вывода 0 - 13).

Таким образом, при необходимости применения дополнительных портов ввода/вывода имеется возможность сконфигурировать неиспользуемые аналоговые входы.

Во время работы с кнопками мы уже познакомились с функцией `digitalRead`, которая умеет считывать цифровой сигнал с определенного вывода контроллера. У этой функции существует аналоговая версия `analogRead`, которая может делать то же самое, но только для аналогового сигнала:

результат = `analogRead(номер_контакта);`

После ее вызова в программе переменная результат будет хранить уровень аналогового сигнала, считанный с соответствующего контакта. При этом, в отличие от цифрового сигнала, функция `analogRead` возвращает число от 0 до 1023, где 0 означает напряжение 0 вольт, а 1023 – 5 вольт. Промежуточным напряжениям будут соответствовать промежуточные значения. Подключать потенциометр надо к аналоговым входам A0..A5 платы Arduino как показано на рисунке:



Напишем программу, которая будет измерять положение потенциометра раз в полсекунды и отправлять его через последовательный порт:

```
const int analog_R = A0;  
void setup() {  
  pinMode(analog_R, INPUT);  
  Serial.begin(9600);  
}  
void loop() {  
  int result = analogRead( analog_R );  
  Serial.println( result );  
  delay(500);  
}
```

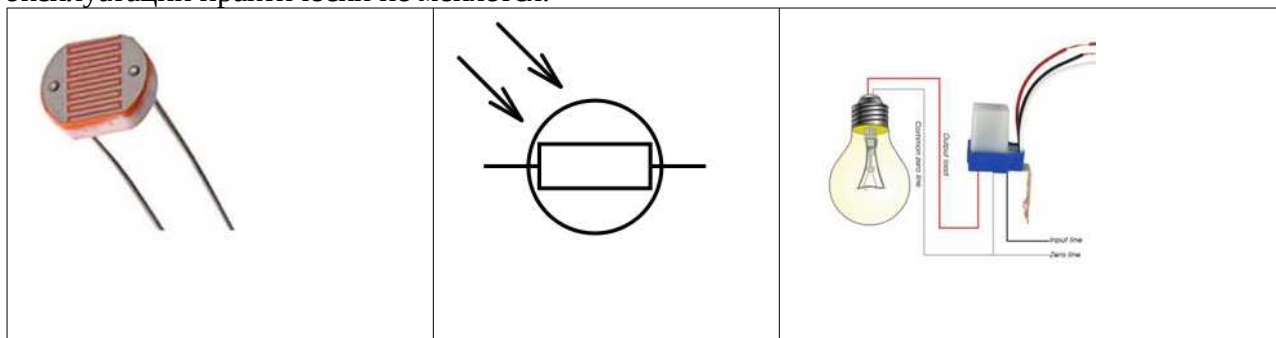
Обратите внимание, что если вы будете использовать несколько аналоговых датчиков, то при последовательном чтении показания с них могут считываться неустойчиво. Это может происходить потому, что датчики влияют друг на друга в момент переключения

Фоторезистор

Датчики освещенности (освещения), построенные на базе фоторезисторов, довольно часто используются в реальных проектах. Они относительно просты, не дороги, их легко найти и купить в любом интернет-магазине. Фоторезистор ардуино позволяет контролировать уровень освещенности и реагировать на его изменение.

Фоторезистор, как следует из названия, имеет прямое отношение к резисторам, которые часто встречаются практически в любых электронных схемах. Основной характеристикой обычного резистора является величина его сопротивления. От него зависят напряжение и ток, с помощью резистора мы выставляем нужные режимы работы других

компонентов. Как правило, значение сопротивления у резистора в одних и тех же условиях эксплуатации практически не меняется.



В отличие от обычного резистора, фоторезистор может менять свое сопротивление в зависимости от уровня окружающего освещения. Это означает, что в электронной схеме будут постоянно меняться параметры, в первую очередь нас интересует напряжение, падающее на фоторезисторе. Фиксируя эти изменения напряжения на аналоговых пинах ардуино, мы можем менять логику работы схемы, создавая тем самым адаптирующиеся под внешние условия устройства.

Фоторезисторы достаточно активно применяются в самых разнообразных системах. Самый распространенный вариант применения — фонари уличного освещения. Если на город опускается ночь или стало пасмурно, то огни включаются автоматически. Можно сделать из фоторезистора экономную лампочку для дома, включающуюся не по расписанию, а в зависимости от освещения. На базе датчика освещенности можно сделать даже охранную систему, которая будет срабатывать сразу после того, как закрытый шкаф или сейф открыли и осветили.

Самый популярный и доступный вариант датчика на рынке — это модели массового выпуска китайских компаний, клоны изделий производителя VT.

VT83N1 — 12-100кОм (12К — освещенный, 100К — в темноте)

VT93N2 — 48-500кОм (48К — освещенный, 100К — в темноте).

Иногда для уточнения информации о моделях продавец предоставляет специальный документ от производителя (даташит). Кроме параметров работы там же указывается точность датчика. У всех моделей диапазон чувствительности расположен в видимой части спектра. Собирая датчик света нужно понимать, что точность срабатывания — понятие условное. Даже у моделей одного производителя, одной партии, одной закупки отличаться она может на 50% и более.

На заводе датчики настраиваются на длину волны от красного до зелёного света. Большинство при этом «видит» и инфракрасное излучение. Особо точные датчики могут улавливать даже ультрафиолет.

Основным недостатком фоторезисторов является чувствительность к спектру. В зависимости от типа падающего света сопротивление может меняться на несколько порядков. К минусам также относится низкая скорость реакции на изменение освещённости. Если свет мигает — датчик не успевает отреагировать. Если же частота изменения довольно велика — резистор вообще перестанет «видеть», что освещённость меняется. К плюсам можно отнести простоту и доступность. Прямое изменение сопротивления в зависимости от попадающего на неё света позволяет упростить электрическую схему подключения. Сам фоторезистор очень дешев, входит в состав многочисленных наборов и конструкторов ардуино, поэтому доступен практически любому начинающему ардуинщику.

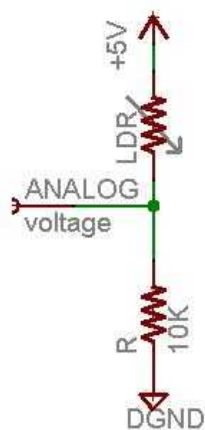


Схема подключения датчика освещенности к ардуино довольно проста. Если мы используем фоторезистор, то в схеме подключения датчик реализован как делитель напряжения. Одно плечо меняется от уровня

освещённости, второе – подаёт напряжение на аналоговый вход. В микросхеме контроллера это напряжение преобразуется в цифровые данные через АЦП. Т.к. сопротивление датчика при попадании на него света уменьшается, то и значение падающего на нем напряжения будет уменьшаться.

В зависимости от того, в каком плече делителя мы поставили фоторезистор, на аналоговый вход будет подаваться или повышенное или уменьшенное напряжение. В том случае, если одна нога фоторезистора подключена к земле, то максимальное значение напряжения будет соответствовать темноте (сопротивление фоторезистора максимальное, почти все напряжение падает на нем), а минимальное – хорошему освещению (сопротивление близко к нулю, напряжение минимальное). Если мы подключим плечо фоторезистора к питанию, то поведение будет противоположным.

Пример скетча датчика освещенности на фоторезисторе

Написать скетч для датчика освещенности довольно просто. Нам нужно только снять текущее значение напряжения с того аналогового пина, к которому подключен датчик. Делается это с помощью известной нам всем функции `analogRead()`. Затем мы можем выполнять какие-то действия, в зависимости от уровня освещенности.

Давайте напишем скетч для датчика освещенности, включающего или выключающего светодиод при определенной освещенности.

Алгоритм работы таков:

- Определяем уровень сигнала с аналогового пина.
- Сравниваем уровень с пороговым значением. Максимально значение будет соответствовать темноте, минимальное – максимальной освещенности. Пороговое значение выберем равное 300.
- Если уровень меньше порогового – темно, нужно включать светодиод.
- Иначе – выключаем светодиод.

```
#define PIN_LED 13
#define PIN_PHOTO_SENSOR A0
void setup() {
  Serial.begin(9600);
  pinMode(PIN_LED, OUTPUT);
}
void loop() {
  int val = analogRead(PIN_PHOTO_SENSOR);
  Serial.println(val);
  if (val < 300) {
    digitalWrite(PIN_LED, LOW);
  } else {
    digitalWrite(PIN_LED, HIGH);
  }
}
```

Прикрывая фоторезистор (руками или светонепроницаемым предметом), можем наблюдать включение и выключение светодиода. Изменяя в коде пороговый параметр, можем заставлять включать/выключать светодиод при разном уровне освещения.

Датчик температуры LM35.



LM35 это прецизионный интегральный датчик температуры с широким диапазоном температур, высокой точностью измерения, калиброванным выходом по напряжению. Именно эти качества определили популярность датчика.

Общая информация.

Серия LM35 это прецизионные интегральные датчики температуры, у которых выходное напряжение пропорционально температуре по шкале Цельсия. Это одно из преимуществ над датчиками с выходным напряжением по шкале Кельвина. Не требуется вычитать высокостабильное напряжение из выходного напряжения для перевода в шкалу по Цельсию.

LM35 обеспечивает измерение температуры с точностью $\pm 0.25\text{ }^{\circ}\text{C}$ в комнатных условиях и с точностью $\pm 0.75\text{ }^{\circ}\text{C}$ в полном диапазоне рабочих температур $-55 \dots +150\text{ }^{\circ}\text{C}$, без внешней калибровки или подгонки выходного напряжения.

Низкая цена датчика объясняется подгонкой и калибровкой датчиков на этапе изготовления.

Низкое выходное сопротивление, линейное значение выходного напряжения и прецизионная калибровка делают датчик LM35 крайне удобным для подключения к измерительным цепям.

Датчик может использоваться как с однополярным напряжением питания, так и с двух полярным.

В связи с тем, что датчик потребляет ток только 60 мкА, у него очень низкий уровень собственного разогрева, менее $0.1\text{ }^{\circ}\text{C}$ при неподвижном воздухе.

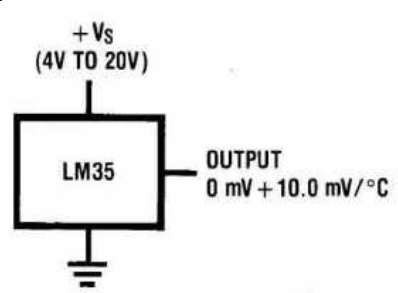
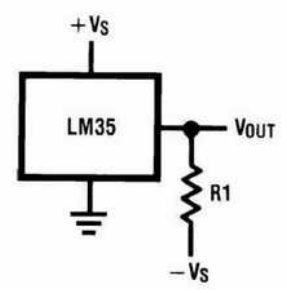
Датчик LM35 допускает работу в диапазоне температур $-55 \dots +150\text{ }^{\circ}\text{C}$, LM35C работает в диапазоне $-40 \dots +110\text{ }^{\circ}\text{C}$ (от $-10\text{ }^{\circ}\text{C}$ с улучшенной точностью).

LM35 выпускается в корпусе TO-46, датчики LM35C, LM35CA и LM35D – в корпусе TO-92. Для LM35D возможны также исполнения в корпусах SO-8 и TO-220.

Особенности датчиков LM35.

- Значение температуры калибровано в шкале Цельсия.
- Линейное значение напряжения на выходе с коэффициентом $10\text{ мВ}/^{\circ}\text{C}$.
- Гарантирована точность $0.5\text{ }^{\circ}\text{C}$ (при $25\text{ }^{\circ}\text{C}$).
- Параметры нормированы для полного диапазона температур $-55 \dots +150\text{ }^{\circ}\text{C}$.
- Удобны для использования в устройствах с удаленным подключением датчиков.
- Низкая цена.
- Работает в широком диапазоне напряжения питания $4 - 30\text{ В}$.
- Потребляемый ток менее 60 мкА .
- Низкий уровень собственного разогрева – $0.08\text{ }^{\circ}\text{C}$ при неподвижном воздухе.
- Нелинейность только $\pm 0.25\text{ }^{\circ}\text{C}$.
- Низкое выходное сопротивление – 0.1 Ом , при токе нагрузки 1 мА .

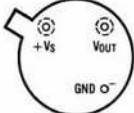

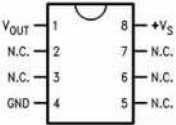
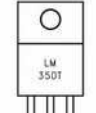
Типовые схемы включения.

<p>Базовая схема включения LM35 в качестве датчика температуры с диапазоном измерения $+2 \dots +150\text{ }^{\circ}\text{C}$.</p> 	<p>Схема включения LM35 в качестве датчика температуры с полным диапазоном измерения $-55 \dots +150\text{ }^{\circ}\text{C}$.</p>  <p>Резистор $R1 = -Vs / 50\text{ мкА}$</p>
---	--

Примерное напряжение на выходе датчика в зависимости от температуры

Vout	Температура
+ 1500 мВ	+ 150 °С
+ 250 мВ	+ 25 °С
- 550 мВ	- 55 °С

Назначение выводов.

Корпус	Датчики	Назначение выводов
TO-46	LM35H, LM35AH, LM35CH, LM35CAH, LM35DH	<p>TO-46 Metal Can Package*</p>  <p>BOTTOM VIEW DS0005516-1</p> <p>*Case is connected to negative pin (GND)</p>
TO-92	LM35CZ, LM35CAZ, LM35DZ	<p>TO-92 Plastic Package</p>  <p>BOTTOM VIEW</p>
SO-8	LM35DM	<p>SO-8 Small Outline Molded Package</p>  <p>N.C. = No Connection DS0005516-21</p> <p>Top View</p>
TO-220	LM35DT	<p>TO-220 Plastic Package*</p>  <p>DS0005516-24</p> <p>*Tab is connected to the negative pin (GND).</p>

Рекомендации по применению.

Использовать датчики LM35 намного проще, чем другие датчики температуры.

Корпуса датчиков должны быть приклеены или прижаты к контролируемой поверхности. Тогда температура датчиков будет в пределах 0.01 °С от температуры поверхности.

Предполагается, что температура окружающего воздуха равна температуре поверхности. В противном случае фактическая температура датчика LM35 будет средней между температурами поверхности и воздуха. Особенно, это имеет значение для пластиковых корпусов TO-92, в которых медные выводы отводят значительное количество тепла. В этом случае реальная температура датчика может быть даже ближе к температуре окружающего воздуха, а не к температуре поверхности.

Чтобы минимизировать эту проблему, старайтесь проводить выводы подключения датчиков так, чтобы их температура была равна температуре поверхности, на которой

установлен датчик. Один из способов – покрыть участок поверхности с выводами LM35 компаундом. Это выровняет температуру выводов датчика и поверхности, и уменьшит влияние окружающего воздуха.

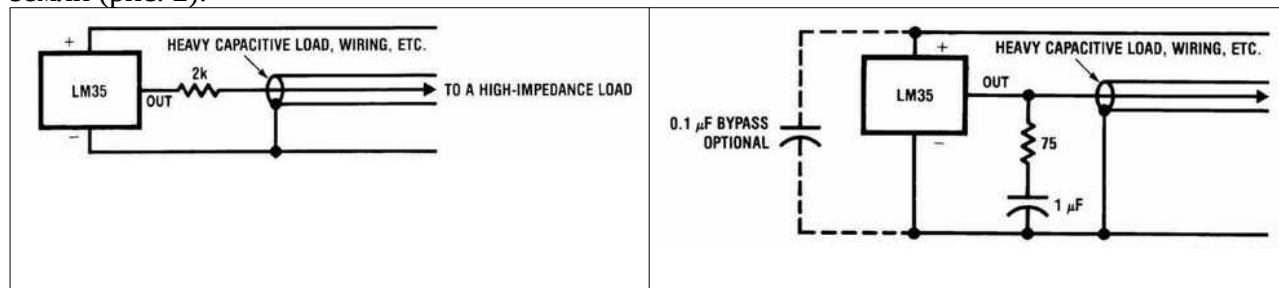
Датчик в корпусе ТО-46 может быть припаян к металлической поверхности или трубке. Повреждения корпуса не допустимы. В этом случае общий вывод датчика будет соединен с металлом.

Как вариант, LM35 может быть установлен на краю герметичной трубки и погружен в ванну или вкручен в отверстие с резьбой в баке.

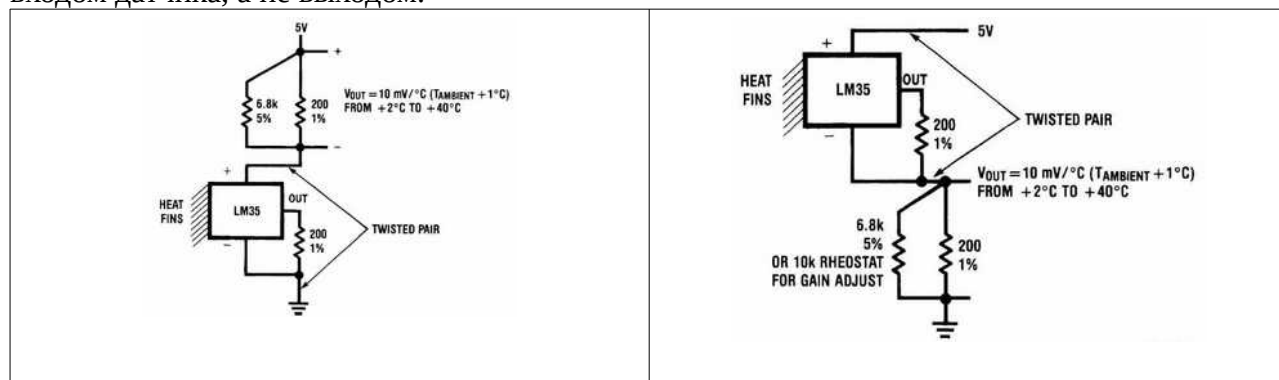
Как в случае применения любой другой интегральной схемы, датчик LM35 и провода к нему должны быть хорошо изолированы, без доступа влаги, во избежание утечки и коррозии. Это особенно важно, если схема работает при низких температурах, вызывающих конденсацию влаги. В этом случае необходимо применять изолирующие лаки и покрытия.

Емкостная нагрузка датчика.

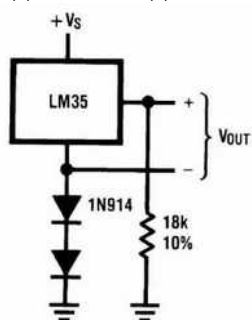
Как и большинство микромощных схем, у LM35 существуют ограничения по емкости нагрузки. Без специальных мер, к выходу LM35 можно подключать нагрузку емкостью не более 50 пкФ. При большей емкости, необходимо развязать емкость резистором (рис. 1). Можно уменьшить влияние емкости демпфирующей R-C цепочкой относительно земли (рис. 2).



В случае применения LM35 с нагрузочным резистором 200 Ом (рис. 3, рис. 4), емкость проводов связи не оказывает влияния, т.к. емкость подключена между землей и входом датчика, а не выходом.



Температурный датчик с одним питанием, -55 ... +150 °C.



Простейшая программа:

```
float tempC;
int reading;
void setup()
{
  analogReference(INTERNAL); // включаем внутренний источник опорного 1,1
  Serial.begin(9600);
}
void loop()
{
  reading = analogRead(A0); // получаем значение с аналогового входа A0
  tempC = reading / 9.31; // переводим в цельсии
  Serial.print(tempC); // отправляем в монитор порта
  Serial.println(" C");
  delay(1000); // ждем секунду
}
```

ВОЛЬТ

Характеристики датчика

Параметр	Условия испытаний	LM35A			LM35CA			UNIT
		TYP	TESTED LIMIT ⁽¹⁾	DESIGN LIMIT ⁽²⁾	TYP	TESTED LIMIT ⁽¹⁾	DESIGN LIMIT ⁽²⁾	
Точность	$T_A = 25^{\circ}\text{C}$	± 0.2	± 0.5		± 0.2	± 0.5		$^{\circ}\text{C}$
	$T_A = -10^{\circ}\text{C}$	± 0.3			± 0.3		± 1	
	$T_A = T_{\text{MAX}}$	± 0.4	± 1		± 0.4	± 1		
	$T_A = T_{\text{MIN}}$	± 0.4	± 1		± 0.4		± 1.5	
Нелинейность	$T_{\text{MIN}} \leq T_A \leq T_{\text{MAX}}$, $-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	± 0.18		± 0.35	± 0.15		± 0.3	$^{\circ}\text{C}$
Усиление	$T_{\text{MIN}} \leq T_A \leq T_{\text{MAX}}$	10	9.9		10		9.9	$\text{mV}/^{\circ}\text{C}$
	$-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	10	10.1		10		10.1	
Нестабильность вых. напряж	$T_A = 25^{\circ}\text{C}$	± 0.4	± 1		± 0.4	± 1		mV/mA
	$T_{\text{MIN}} \leq T_A \leq T_{\text{MAX}}$, $-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	± 0.5		± 3	± 0.5		± 3	
Потребляемый ток	$V_S = 5\text{ V}$, 25°C	56	67		56	67		μA
	$V_S = 5\text{ V}$, $-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	105		131	91		114	
	$V_S = 30\text{ V}$, 25°C	56.2	68		56.2	68		
	$V_S = 30\text{ V}$, $-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	105.5		133	91.5		116	
Изменение тока покоя	$4\text{ V} \leq V_S \leq 30\text{ V}$, 25°C	0.2	1		0.2	1		μA
	$4\text{ V} \leq V_S \leq 30\text{ V}$, $-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	0.5		2	0.5		2	
Темпер. коэфф. тока покоя	$-40^{\circ}\text{C} \leq T_J \leq 125^{\circ}\text{C}$	0.39		0.5	0.39		0.5	$\mu\text{A}/^{\circ}\text{C}$

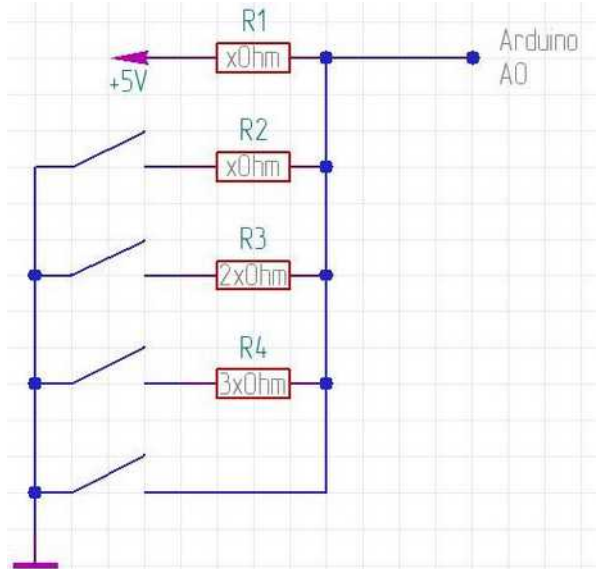
Использование аналоговых входов для подключения кнопок.

Задача с подключением нескольких кнопок на один вход Arduino наверняка заинтересует всех пользователей этого аналого-цифрового преобразователя. Если каждый из восьми портов будет занят только одной кнопкой, то это существенно сократит эффективность системы.

К аналоговому входу можно подключить большое количество кнопок. Принцип работы схем основан на чтении и интерпретации аналого-цифровым преобразователем микроконтроллера индивидуального напряжения, формируемого разными комбинациями отдельных участков схемы.

На своем аналоговом входе Arduino производит измерение напряжения, преобразуя его в числа от 0 до 1023. Задача состоит в том, чтобы на входе A0 с помощью резистора создать максимальное напряжение +5В, благодаря этому преобразователь будет постоянно

считывать с него крайнее число 1023. К этому же аналоговому входу выполняем подключение кнопки, заземляя ее.



сказать, какая из кнопок нажата.

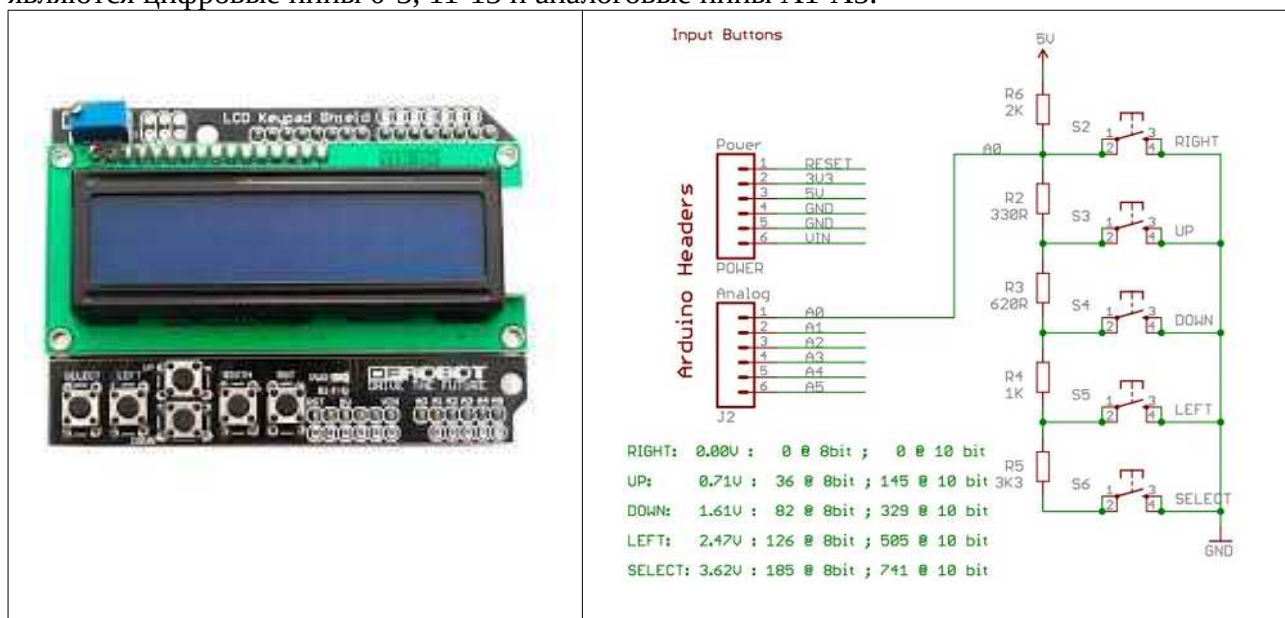
Результатом будет постоянное значение 0. Добившись крайних значений можно экспериментировать с другими кнопками, подключая их с одного конца к земле, а с другого – ко входу через резистор с определенным сопротивлением. Разное сопротивление в итоге даст нужное значение.

На принципиальной схеме видно, что нажимая кнопку – мы формируем делитель напряжения, благодаря чему Arduino считывает данные с нижнего резистора.

Кнопки будут иметь различные значения напряжения, так как в делителе будет сумма сопротивления питающей ветви и ветви определенной кнопки. В результате меняющееся значение сопротивления позволяет точно

LCD Keypad Shield

Шилд представляет собой плату с встроенными модулями индикации и управления. Индикация осуществляется с помощью LCD-дисплея TC1602, управление – через встроенные кнопки. Есть возможность регулировки яркости дисплея прямо на плате с помощью подстроечного резистора. Плата снабжена разъемами, в которые могут быть подключены другие устройства, например, датчики. Для работы с экраном используются пины 4-10, для определения нажатия кнопок – только один аналоговый пин A0. Свободными являются цифровые пины 0-3, 11-13 и аналоговые пины A1-A5.



На плате присутствуют пять управляющих кнопок, работа с которыми ведется через один аналоговый пин A0. В шилде использован достаточно распространенный способ простого кодирования сигнала, при котором каждая кнопка формирует определенное значение напряжения, которое после АЦП преобразуется в соответствующее значение от 0 до 1023. Таким образом, мы можем передавать информацию о нажатии разных кнопок через один пин, считывая его при помощи функции `analogRead()`;

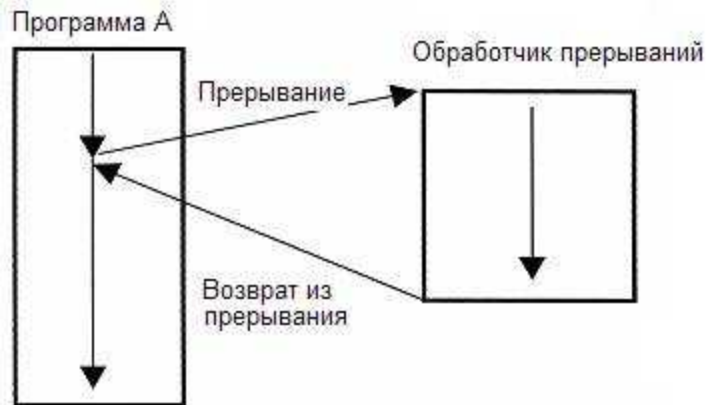
Значения уровня сигнала на пине A0 в зависимости от выбранной кнопки:

Нажатие кнопки	Значение на аналоговом пине
RIGHT	0-100
UP	100-200
DOWN	200-400
LEFT	400-600
SELECT	600-800
Клавиша не нажата	800-1023

В выбранном методе кодирования есть два главных недостатка:
Нельзя отслеживать одновременное нажатие нескольких кнопок;
Возможные искажения сигнала могут привести к ложным срабатываниям.

Прерывания в Ардуино

Прерывания – очень важный механизм Arduino, позволяющий внешним устройствам взаимодействовать с контроллером при возникновении разных событий. Установив обработчик аппаратных прерываний в скетче, мы сможем реагировать на включение или выключение кнопки, нажатие клавиатуры, мышки, тики таймера RTC, получение новых данных по UART, I2C или SPI.



Прерывание – это сигнал, который сообщает процессору о наступлении какого-либо события, которое требует незамедлительного внимания. Процессор должен отреагировать на этот сигнал, прервав выполнение текущих инструкций и передав управление обработчику прерывания (ISR, Interrupt Service Routine). Обработчик – это обычная функция, которую мы пишем сами и помещаем туда тот код, который

должен отреагировать на событие.

После обслуживания прерывания ISR функция завершает свою работу и процессор с удовольствием возвращается к прерванным занятиям – продолжает выполнять код с того места, в котором остановился. Все это происходит автоматически, поэтому наша задача заключается только в том, чтобы написать обработчик прерывания, ничего при этом не сломав и не заставляя процессор слишком часто отвлекаться на нас. Понадобится понимание схемы, принципов работы подключаемых устройств и представление о том, как часто может вызываться прерывание, каковы особенности его возникновения. Все это и составляет основную сложность работы с прерываниями.

Аппаратные и программные прерывания

Прерывания в Ардуино можно разделить на несколько видов:

Аппаратные прерывания. Прерывание на уровне микропроцессорной архитектуры. Самое событие может произойти в произвольный момент от внешнего устройства – например, нажатие кнопки на клавиатуре, движение компьютерной мыши и т.п.

Программные прерывания. Запускаются внутри программы с помощью специальной инструкции. Используются для того, чтобы вызвать обработчик прерываний.

Внутренние (синхронные) прерывания. Внутреннее прерывание возникает в результате изменения или нарушения в исполнении программы (например, при обращении к недопустимому адресу, недопустимый код операции и другие).

Зачем нужны аппаратные прерывания

Аппаратные прерывания возникают в ответ на внешнее событие и исходят от внешнего аппаратного устройства. В Ардуино представлены 4 типа аппаратных прерываний. Все они различаются сигналом на контакте прерывания:

- Контакт притянут к земле. Обработчик прерывания выполняется до тех пор, пока на пине прерывания будет сигнал LOW.
- Изменение сигнала на контакте. В таком случае Ардуино выполняет обработчик прерывания, когда на пине прерывания происходит изменение сигнала.
- Изменение сигнала от LOW к HIGH на контакте – при изменении с низкого сигнала на высокий будет выполняться обработчик прерывания.
- Изменение сигнала от HIGH к LOW на контакте – при изменении с высокого сигнала на низкий будет выполняться обработчик прерывания.

Прерывания полезны в программах Ардуино, так как помогают решать проблемы синхронизации. Например, при работе с UART прерывания позволяют не отслеживать поступление каждого символа. Внешнее аппаратное устройство подает сигнал прерывания, процессор сразу же вызывает обработчик прерывания, который вовремя захватывает символ. Это позволяет экономить процессорное время, которое без прерываний тратилось бы на проверку статуса UART, вместо этого все необходимые действия выполняются обработчиком прерывания, не затрагивая главную программу. Особых возможностей от аппаратного устройства не требуется.

Основными причинами, по которым необходимо вызвать прерывание, являются:

- Определение изменения состояния вывода;
- Прерывание по таймеру;
- Прерывания данных по SPI, I2C, USART;
- Аналогово-цифровое преобразование;
- Готовность использовать EEPROM, флеш-память.

Как реализуются прерывания в Ардуино

При поступлении сигнала прерывания работа в цикле loop() приостанавливается. Начинается выполнение функции, которая объявляется на выполнение при прерывании. Объявленная функция не может принимать входные значения и возвращать значения при завершении работы. На сам код в основном цикле программы прерывание не влияет. Для работы с прерываниями в Ардуино используется стандартная функция attachInterrupt().

Отличие реализации прерываний в разных платах Ардуино

Система	int0	int1	int2	int3	int4	int5
ARDUINO UNO	pin 2	pin 3				
ARDUINO MEGA	pin 2	pin 3	pin 21	pin 20	pin 19	pin 18

В зависимости от аппаратной реализации конкретной модели микроконтроллера есть несколько прерываний. Плата Arduino Uno имеет 2 прерывания на втором и третьем пине, но если

требуется более двух выходов, плата поддерживает специальный режим «pin-change». Этот режим работает по изменению входа для всех пинов. Отличие режима прерывания по изменению входа заключается в том, что прерывания могут генерироваться на любом из восьми контактов. Обработка в таком случае будет сложнее и дольше, так как придется отслеживать последнее состояние на каждом из контактов.

На других платах число прерываний выше. Например, плата Ардуино Мег 2560 имеет 6 пинов, которые могут обрабатывать внешние прерывания. Для всех плат Ардуино при работе с функцией attachInterrupt (interrupt, function, mode) аргумент Interrupt 0 связан с цифровым пином 2.

Прерывания в языке Arduino

Функция `attachInterrupt` используется для работы с прерываниями. Она служит для соединения внешнего прерывания с обработчиком.

Синтаксис вызова: **`attachInterrupt(interrupt, function, mode)`**

Аргументы функции:

`interrupt` – номер вызываемого прерывания (стандартно 0 – для 2-го пина, для платы Ардуино Уно 1 – для 3-го пина),

`function` – название вызываемой функции при прерывании (важно – функция не должна ни принимать, ни возвращать какие-либо значения),

`mode` – условие срабатывания прерывания.

Возможна установка следующих вариантов условий срабатывания:

LOW – выполняется по низкому уровню сигнала, когда на контакте нулевое значение. Прерывание может циклично повторяться – например, при нажатой кнопке.

CHANGE – по фронту, прерывание происходит при изменении сигнала с высокого на низкий или наоборот. Выполняется один раз при любой смене сигнала.

RISING – выполнение прерывания один раз при изменении сигнала от LOW к HIGH.

FALLING – выполнение прерывания один раз при изменении сигнала от HIGH к LOW.

Важные замечания

При работе с прерываниями нужно обязательно учитывать следующие важные ограничения:

Функция – обработчик не должна выполняться слишком долго. Все дело в том, что Ардуино не может обрабатывать несколько прерываний одновременно. Пока выполняется ваша функция-обработчик, все остальные прерывания останутся без внимания и вы можете пропустить важные события. Если надо делать что-то большое – просто передавайте обработку событий в основном цикле `loop()`. В обработчике вы можете лишь устанавливать флаг события, а в `loop` – проверять флаг и обрабатывать его.

Нужно быть очень аккуратными с переменными. Интеллектуальный компилятор C++ может “пере оптимизировать” вашу программу – убрать не нужные, на его взгляд, переменные. Компилятор просто не увидит, что вы устанавливаете какие-то переменные в одной части, а используете – в другой. Для устранения такой вероятности в случае с базовыми типами данных можно использовать ключевое слово `volatile`, например так: `volatile boolean state = 0`. Но этот метод не работает со сложными структурами данных. Так что надо быть всегда на чеку.

Не рекомендуется использовать большое количество прерываний (старайтесь не использовать более 6-8). Большое количество разнообразных событий требует серьезного усложнения кода, а, значит, ведет к ошибкам. К тому же надо понимать, что ни о какой временной точности исполнения в системах с большим количеством прерываний речи быть не может – вы никогда точно не поймете, каков промежуток между вызовами важных для вас команд.

В обработчиках категорически нельзя использовать `delay()`. Механизм определения интервала задержки использует таймеры, а они тоже работают на прерываниях, которые заблокирует ваш обработчик. В итоге все будут ждать всех и программа зависнет. По этой же причине нельзя использовать протоколы связи, основанные на прерываниях (например, i2c).

Задания к лабораторной работе №3

1. Чтение аналогового входа, изменение аналогового входа потенциометром
2. Чтение температуры с датчика LM35, вывод на консоль.
3. Чтение датчика освещенности с выводом датчика на консоль,.
4. Калибровка датчика температуры, внесение данных калибровки в программу через консоль.

Дополнительные задания

5. Простейший датчик освещенности — включение светодиода при недостаточной освещенности. Установка порога срабатывания датчика через консоль. Кнопка - включение/отключение датчика. Отдельный светодиод показывает включение датчика (алгоритма работы) Вариант установки датчика освещенности через потенциометр.

Задание для дистанционного обучения

Разработать программу простейшего терморегулятора, со следующими характеристиками:

1. В качестве датчика температуры использовать датчик LM35 (A2). Текущая температура T_t . Точность расчетов и хранения температуры не ниже 0.1 градуса.
2. Установку целевой температуры (T_c) проводить с помощью переменного резистора присоединенного в выходу A0. Точность установки до 0.1 градуса. Диапазон регулирования +2 ... +35 градусов.
3. Считать нагревательным элементом светодиод красного цвета (D12).
4. Терморегулятор должен иметь гистерезис (G_c), при этом температура включения нагревательного элемента равна целевая температура минус гистерезис ($вкл = T_c - G_c$), а выключения — целевая температура ($выкл = T_c$). Гистерезис (G_c) по умолчанию 2 градуса
5. Кнопка D2 включает и выключает работу (регулирование) терморегулятора. При включении работы загорается светодиод синего цвета (D13).
6. Добавить отображение текущего состояния регулятора, используя трех цветный светодиод (D9-D11). Отображать следующие состояния:
 - Регулирование выключено — все светодиоды выключены.
 - ($T_t < T_c - G_c$) — горит красный светодиод (D9)
 - ($T_t \geq T_c - G_c$) & ($T_t \leq T_c$) — горит зеленый светодиод (D10)
 - ($T_t > T_c$) — горит синий светодиод (D11)

Для продвинутых пользователей.

7. Добавить выполнение через консоль следующих команд:

state — вывод состояния терморегулятора. Выводится:

- текущая температура
- целевая температур
- гистерезис
- вкл/вкл нагревателя (D12)
- вкл/вкл всего устройства (D13)
- систематическая ошибка датчика.

setGT — ввод значения гистерезиса, разрешены значения -5 ... +5 с разрешением 1 градус (только целые значения).

setERR — ввод систематической ошибки датчика LM35. Ошибка ДОБАВЛЯЕТСЯ к текущему значению температуры. Разрешенные значения -4 ... +4 с разрешением 1 градус (только целые значения).

ON — включить регулирование, если оно включено то сообщить об этом. Не забываем о D13

OFF — выключить регулирование, если оно выключено то сообщить об этом. Не забываем о D13

help — вывести список команд.