Привет ребята, в этом выпуске я расскажу о том о чем забыл упомянуть в прошлом уроке про типы данных и переменные. Так что это продолжение первого урока, урок номер 1.1.

Начнем с того что многие даже не поняли что такое переменная и зачем она нужна. Я не думал что может быть так плохо. Ну ладно. Очень грубо говоря переменная это слово, которое хранит в себе численное значение. Зачем оно нужно? Да просто для удобства и для некоторых расчетов.

Что касается удобства. Например я подключил датчик звука к 5 порту. И чтобы в дальнейшем коде при работе с датчиком не писать везде цифру 5, я в самом начале программы объявляю переменную и называю ее так чтобы название говорило само за себя. Саунд сенсор пин. Теперь там где мне нужно обратиться к пину датчика, я пишу саунд сенсор пин, и уже не держу в голове что датчик подключен в 5 порт. Еще это удобно тем что если я вдруг решу подключить датчик в другой порт, то мне нужно будет изменить это в коде только а одном месте, а не лазить искать где я там обращался к датчику. Или например вы получаете значение с датчика, и хотите использовать его несколько раз в разных местах. Записав значение в переменную, вы сможете спокойно работать дальше. Также понятное название переменной делает код более читаемым как для самого себя так и для других.

Что касается расчетов. Операции с переменными ничем не отличаются от операций с числами, то есть переменные можно складывать вычитать делить и умножать как между собой, так и с обычными числами. Здесь работают законы математики и скобки, ну то есть действия в скобках будут выполнены первыми, как вы должны помнить со школы. Также я привожу несколько математических операторов, просто запомните их, думаю объяснять здесь ничего не нужно.

Следующий момент : переменные можно перезаписывать. Код выполняется сверху вниз и переменная будет принимать новые значения.

Вот смотрите, значение переменной можно увеличить или уменьшить вот таким образом. То есть переменная перезаписыватся с использованием своего старого значения. Это абсолютно легально и отлично работает. Точно также можно делить и умножать и вообще все остальные действия из списка справа тоже можно таким образом применить. Но язык с++ имеет для вас еще пару фишек. Например действие со сложением можно сократить вот таким образом, где += является стандартным оператором. Вот вам остальные операторы для сокращенных действий. Использовать их необязательно, но они делают ваш код более красивым, более эффективным, чуточку быстрее работающим и вызывающим меньшее отвращение у тру программистов.

Еще нужно сказать про константы. Они задаются точно также как переменные, но перед типом данных ставится слово конст. Константа свое значение не меняет, а если вы захотите ее изменить то компилятор выдаст ошибку, во, присвоение переменной, предназначенной только для чтения.

Следующий момент, область видимости переменных. Переменные есть двух типов, локальные и глобальные. Глобальные объявляются вот здесь, вне функций. Переменная объявленная в коде прошивки вне функций будет доступна везде, во всех функциях . То есть мы можем пользоваться нашей переменной сенсор пин в любом месте, в функции сетап, в функции луп, и в любой функции внутри функции. Видите, программа компилируется.

Если объявить переменную внутри функции, то она будет доступна только внутри этой функции, и нигде больше. Например я объявлю сенсор пин в функции сетап. Внутри функции сетап я могу пользоваться этой переменной. Но если я попытаюсь обратиться к этой переменной в другой функции, пусть в луп, то будет ошибка, так как переменная локальная и работает только для сетап.

Но! Если я объявлю эту переменную еще раз прямо здесь, то у нас будет две переменные с одинаковым именем но разными значениями, причем каждое значение доступно только в той функции, где переменная была объявлена.

Давайте еще разок. К переменной, объявленной вне функций, можно обращаться из любого места программы, из любой функции, эта переменная называется глобальной. Если переменная объявлена внутри функции, то обратиться к ней можно только внутри этой функции, и такая переменная называется локальной.

Теперь вернемся в самое начало, где я давал имя пину датчика звука. Смотрите в чём дело, я создаю переменную, которая хранит номер Пина. Переменная типа байт, занимает в памяти микроконтроллера 1 байт. Да, эта переменная хранится в памяти как отдельный кусочек со своим именем. В ардуино нано на атмеге328 у нас 32 килобайта памяти для прошивки. Вроде бы один байт незначителен по сравнению с общим объемом памяти, но все равно лишнее место занимает. К чему я об этом?

Есть еще один вариант дать имя какому то значению, это директива дефайн. Работает это вот так: дефайн, имя, значение, трчка запятой кстати не ставится. Теперь в коде мы опять же можем использовать это имя как число. Но отличие от переменной здесь в том, что переменная хранится в памяти микроконтроллера, а когда мы работаем через дефайн, то на стадии компиляции программы все слова в коде прошивки, определенные дефайном тупо заменяются на указаные цифры, а затем программа загружается в ардуино. Наглядно это выглядит вот так.

То есть отдельного места в памяти под хранение этого значения не занимается, и лишнее время на обращение по имени тоже не тратится. В итоге имеем чуть меньший вес и чуууть чуть возросшую скорость работы и опять же одобрительный кивок со стороны бородатых программистов. Использовать директиву дефайн не обязательно, но желательно, если вы хотите написать максимально эффективный код. Нужно помнить что имя объявленное через дефайн это не переменная, и работать с ней нужно как с константой. То есть только читать и не перезаписывать. Дефайном имеет смысл объявлять такие вещи как номер Пина, константы, например время для таймера, или какие нибудь физические или математические константы. Я надеюсь вы поняли в чём соль и особенность этой штуки.

На этом у меня всё, спасибо за внимание, увидимся в следующем уроке.