

Министерство образования Республики Беларусь
Учреждение образования
«Гомельский государственный технический университет
имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ПРАКТИКУМ ПО ВЫПОЛНЕНИЮ
ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ
«ПРОГРАММИРОВАНИЕ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ
НА ОСНОВЕ ОДНОПЛАТНЫХ КОМПЬЮТЕРОВ»
для студентов специальности
1-40 05 01 «Информационные системы и технологии
(по направлениям)»

Соболев Д.В.

Гомель 2019

Содержание

1	Работа с датчиками на Arduino.	4 ч.
2	Подключение датчиков к модулю EV3.	4 ч.
3	Программирование роботов из стандартных наборов на Arduino.	2 ч.
4	Программирование роботов из стандартных наборов на Lego EV3.	2 ч.
5	Работа с периферийных устройств с одноплатными ЭВМ.	2 ч.
6	Программирование движения роботов по заданной траектории.	1 ч.
7	Подключение приводов к одноплатным ЭВМ.	2 ч.

«Работа с датчиками на Arduino» (4 часа)

Получить навыки работы с платами семейства Arduino, научиться подключать различные датчики, устройства вывода, сигнальные и исполнительные устройства.

2.1 Мигающий светодиод

- плата Arduino;
- breadboard;
- 2 провода «папа-папа»;
- светодиод;
- резистор.

2.1.1 Собрать схему подключения светодиода на Arduino (рисунок 1)

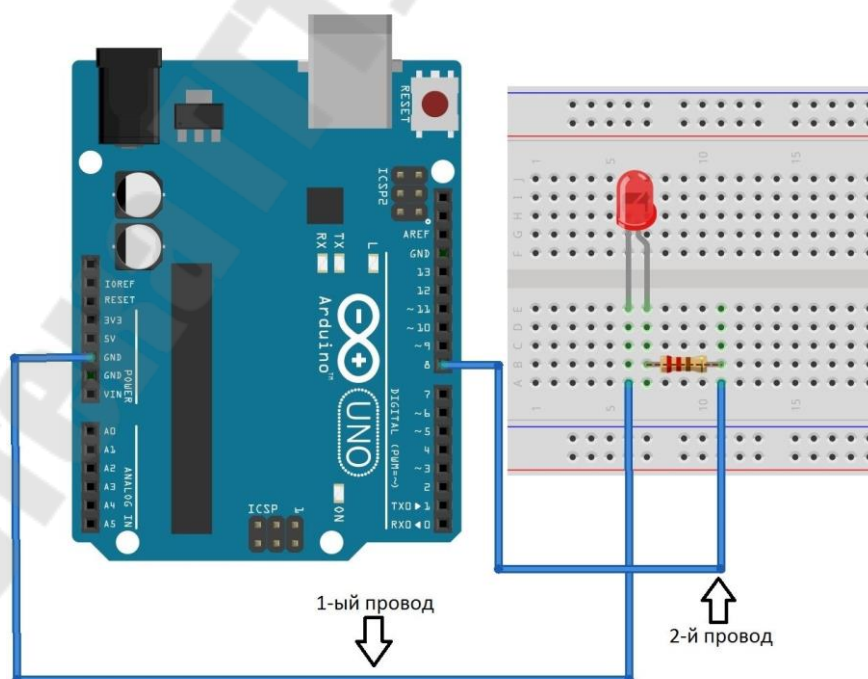


Рисунок 1 – Схема подключения светодиода к плате

2.1.2 Подключить плату к компьютеру, запустить IDE и инициализировать плату (в случае, если не произошло автоматически).

2.1.3 Загрузить или набрать текст программы:

```
int led = 8; //объявление переменной целого типа, содержащей номер порта
```

к которому мы подключили второй провод

```
void setup() //обязательная процедура setup, запускаемая в начале
```

программы; объявление процедур начинается словом void

```
{
```

```
pinMode(led, OUTPUT); //объявление используемого порта, led - номер порта,
```

второй аргумент - тип использования порта - на вход (INPUT) или на выход

```
(OUTPUT)
```

```
}
```

```
void loop() //обязательная процедура loop, запускаемая циклично после
```

процедуры setup

```
{
```

```
digitalWrite(led, HIGH); //эта команда используется для включения или
```

выключения напряжения на цифровом порте; led - номер порта, второй

аргумент - включение (HIGH) или выключение (LOW)

```
delay(1000); //эта команда используется для ожидания между действиями,
```

аргумент - время ожидания в миллисекундах

```
digitalWrite(led, LOW);
```

```
delay(1000);
```

```
}
```

2.1.4 Загрузить скетч на плату и проверить работоспособность.

2.2 Датчик температуры/влажности

Для сборки проекта, описанного в этом уроке, понадобятся следующие детали:

- плата Arduino ;
- датчик DHT11 или DHT22;
- Breadboard;
- провода;
- резистор на 10 кОм.

Две версии сенсоров DHT похожи друг на друга и имеют одинаковую распиновку. Их отличия в характеристиках. Спецификации:

Сенсор DHT11:

- определение влажности в диапазоне 20-80%;
- определение температуры от 0°C до +50°C;
- частота опроса 1 раз в секунду.

Сенсор DHT22:

- определение влажности в диапазоне 0-100%;
- определение температуры от -40°C до +125°C;
- частота опроса 1 раз в 2 секунды.

Подключение сенсоров DHT к Arduino

Датчики DHT имеют стандартные выводы и их просто установить на breadboard.

Датчики DHT имеют 4 вывода:

1. питание.
2. вывод данных
3. не используется.
4. GND (земля).

Между выводами питания и вывода данных нужно разместить резистор номиналом 10 кОм.

Датчик DHT часто продается в виде готового модуля. В этом случае он имеет три вывода и подключается без резистора, т.к. резистор уже есть на плате.

2.2.1 Собрать схему (рисунок 2)

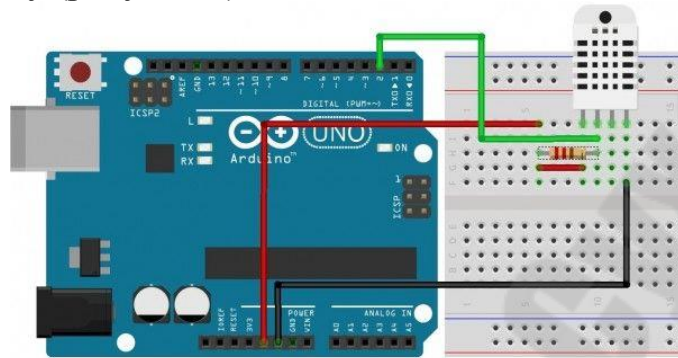


Рисунок 2 – Схема подключения датчика температуры\влажности

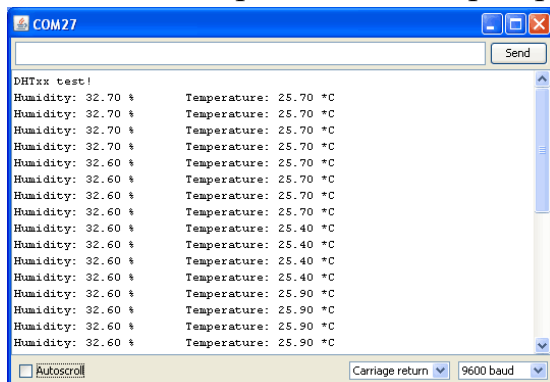
2.2.2 Подключить плату к компьютеру, запустить IDE и инициализировать плату (в случае, если не произошло автоматически).

2.2.3 Набрать текст программы

```
#include "DHT.h"
#define DHTPIN 2 // номер пина, к которому подсоединен датчик
// Раскомментируйте в соответствии с используемым датчиком
// Иницизируем датчик
DHT dht(DHTPIN, DHT22);
//DHT dht(DHTPIN, DHT11);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  // Задержка 2 секунды между измерениями
  delay(2000);
  //Считываем влажность
  float h = dht.readHumidity();
  // Считываем температуру
  float t = dht.readTemperature();
  // Проверка успешно прошло ли считывание.
  if (isnan(h) || isnan(t)) {
    Serial.println("Не удастся считать показания");
    return;
  }
  Serial.print("Влажность: "+h+" %\t"+"Температура: "+t+" °C ");
}
```

«Закомментировать» строчку с лишним датчиком.

2.2.4 Загрузите скетч в контроллер и проверьте правильность работы при помощи Сервис->Монитор порта:



2.3 Подключение сервопривода

Для сборки проекта, описанного в этом уроке, понадобятся следующие детали:

- плата Arduino;
- 3 провода “папа-папа”;
- Сервопривод.

2.3.1 Изучить работу сервопривода.

2.3.2 Собрать схему согласно рисунка 3

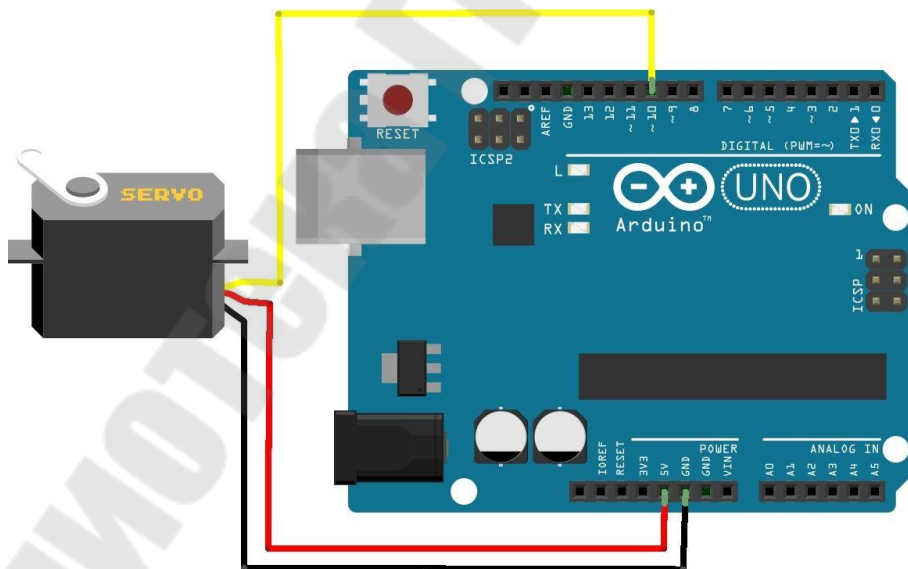


Рисунок 3 – Схема подключения сервопривода

2.3.3 Подключить плату к компьютеру, запустить IDE и инициализировать плату (в случае, если не произошло автоматически).

2.3.4 Набрать текст программы

```
#include <Servo.h> //используем библиотеку для работы с сервоприводом
Servo servo; //объявляем переменную servo типа Servo
void setup() //процедура setup
{
  servo.attach(10); //привязываем привод к порту 10
}
void loop() //процедура loop
{
  servo.write(0); //ставим вал под 0
  delay(2000); //ждем 2 секунды
  servo.write(180); //ставим вал под 180
  delay(2000); //ждем 2 секунды
}
```

2.3.5 Загрузить скетч в плату и проверить работоспособность

3 Задание

Изучить теоретические сведения. Собрать схему в соответствии с заданием. Написать код скетча. Проверить работоспособность.

3.1 «Гирлянда»

Подключить к плате 4 светодиода, написать программу реализующую их работу в режиме гирлянды (уточнить у преподавателя вариант).

3.2 «Метеостанция»

3.2.1 Изучить правила подключения 2хстрочного жидкокристаллического дисплея/или матрицы семисегментных индикаторов к плате.

3.2.2 Повторить задачу пункта 2.2 с датчиком температуры/влажности, но с выводом показателей на индикатор (по варианту).

3.2.3 Изучить принцип работы кнопки и доработать предыдущую задачу на предмет обновления показаний «по нажатию кнопки».

3.3 Управление двигателем

В задачу с сервоприводом добавить функции запуска по нажатию кнопки (без фиксации) и выводом на экран количества проделанных оборотов (по вариантам; со сбросом счетчика и без).

Оформить отчет:

Титульник, цель, ход работы: задание, схема, скетч; выводы.

Лабораторная работа № 2

«Работа с датчиками EV3» (часть 1 2 часа)

1 Цель работы

Получить навыки работы с модулем Lego Mindstorms EV3, научиться подключать различные датчики, устройства вывода, сигнальные и исполнительные устройства.

2 Теоретические сведения

- 2.1 Изучить документацию по контроллеру Lego EV3;
- 2.2 Собрать на практическом занятии робота-тележку;
- 2.3 Изучить принцип работы тактильного датчика (кнопки) и датчика цвета.

3 Задание

3.1 Датчик касания

В состав конструктора Lego mindstorms EV3 входят различные датчики. Главная задача датчиков - представлять информацию из внешней среды модулю EV3, а задача программиста - научиться получать и обрабатывать эту информацию, подавая необходимые команды моторам робота. На протяжении ряда уроков мы будем последовательно знакомиться со всеми датчиками, входящими и в домашний, и в образовательный наборы, научимся взаимодействовать с ними и решать наиболее распространенные задачи управления роботом.

3.1.1 Изучаем первый датчик – датчик касания

Для подключения датчиков к модулю EV3 предназначены порты, обозначенные цифрами "1", "2", "3" и "4". Таким образом, к одному модулю EV3 одновременно можно подключить до четырех различных датчиков. Все порты абсолютно равнозначны и вы можете подключать датчики к любым портам, главное - будьте внимательны при указании номера порта для соответствующих датчиков в ваших программах.

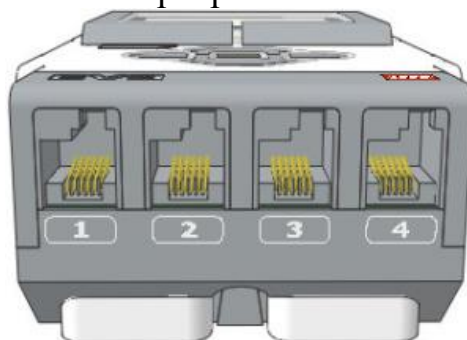


Рисунок 1 – Порты для подключения датчиков

Первым датчиком, который мы изучим, будет датчик касания (Рисунок 2).

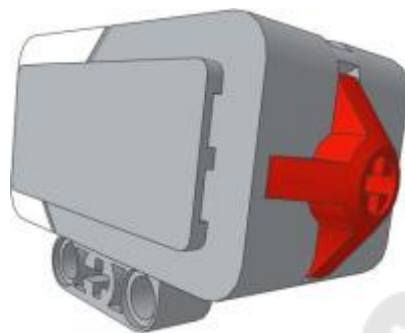


Рисунок 2 – Датчик касания (кнопка)

Этот датчик, по сути, представляет собой специальную кнопку, которая может находиться в двух состояниях: "Нажатие" (Рисунок 3 поз. 1) или "Освобождение" (Рисунок 3 поз. 2). Также, последовательный переход в состояние "Нажатие", а затем "Освобождение" называется: "Щелчок" (Рисунок 3 поз. 3) и может обрабатываться программой, как самостоятельное событие.



Рисунок 3 – Состояния датчика касания

3.1.2 Оранжевая палитра – Управление операторами

Какие же инструменты представляет нам среда программирования для получения информации с датчиков и реагирования на эту информацию в программе? Давайте начнем знакомиться с программными блоками, расположенными в Оранжевой палитре, которая называется "Управление операторами". (Рисунок 4)

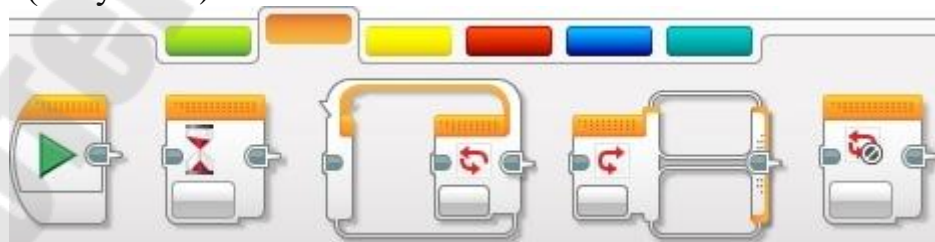


Рисунок 4 – Палитра «Управление операторами»

Программные блоки Оранжевой палитры, не смотря на свою малочисленность, очень важны! С помощью этих блоков мы можем обрабатывать массу событий и условий и сложно представить практическую программу, которая может обойтись без этих блоков.

С самым первым блоком Оранжевой палитры мы уже с вами знакомы: он называется "Начало". Именно с него начинаются все программы для роботов.

Второй программный блок называется "Ожидание". Этот блок заставляет программу ожидать выполнения какого-либо условия или наступления какого-либо события. Пока не выполнится условие, установленное в этом блоке, программа не перейдет к выполнению следующих программных блоков! Если перед тем, как начнется выполнение блока "Ожидание" были включены, какие-либо моторы, то они будут продолжать вращаться с установленной скоростью.

Третий программный блок называется "Цикл". Этот блок многократно выполняет программные блоки, вложенные внутрь его, пока не будет выполнено условие завершения цикла, заданное в настройках блока.

Следующий программный блок называется "Переключатель". Он служит для того, чтобы в зависимости от заданных условий - выполнить одну последовательность программных блоков, вложенных в один из своих контейнеров.

Заключительный программный блок называется "Прерывание цикла". Его предназначение - досрочное прекращение выполнения заданного цикла.

Программные блоки "Ожидание", "Цикл" и "Переключатель" имеют множество режимов и соответствующих настроек, знакомиться с которыми мы будем на практических примерах, последовательно и с наглядными пояснениями.

3.1.3. Оранжевая палитра, программный блок "Ожидание"

Перед тем, как приступить к решению практических задач, давайте закрепим датчик касания на нашем роботе, как показано на Рисунок 5, и подключим его кабелем к порту "1" модуля EV3.

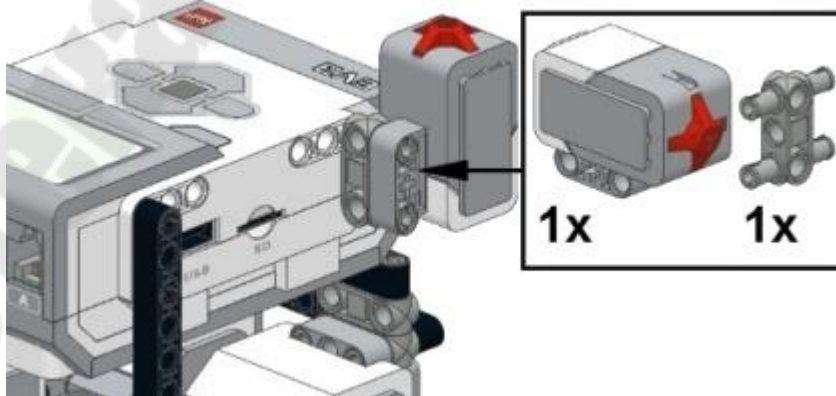


Рисунок 5 – Размещение кнопки на тележке

Задача 3.2: необходимо написать программу, запускающую движение робота по щелчку кнопки.

Решение:

Само условие задачи подсказывает нам возможное решение: перед началом движения - необходимо дождаться нажатия-отпускания кнопки датчика касания. Возьмем программный блок "Ожидание", изменим режим программного блока на "Датчик касания" - "Сравнение" (Рисунок 6).

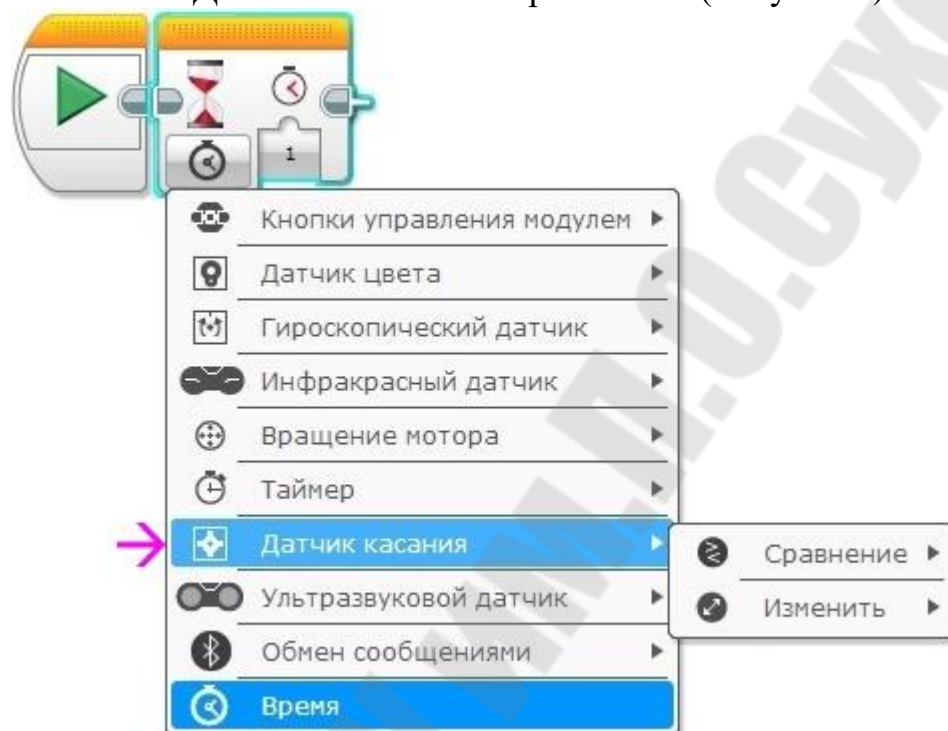


Рисунок 6 – Режим «сравнение»

Как можно увидеть - программный блок "Ожидание" сменил свое отображение! Рядом с песочными часами появилось изображение датчика касания (Рисунок 7 поз. 1), помогающее в программе визуально оценивать установленный режим работы. Настройка программного блока "Состояние" задает требуемое состояние датчика, достижение которого прекратит выполнение блока "Ожидание" (Рисунок 7 поз. 2). Настройка "Состояние" может принимать следующие значения: "0" - "Отпущено", "1" - "Нажатие", "2" - "Щелчок". Для решения нашей задачи выберем состояние "Щелчок". Вывод "Измеренное значение" (Рисунок 7 поз. 3) при необходимости позволяет передать окончательное состояние датчика для обработки в другой программный блок.

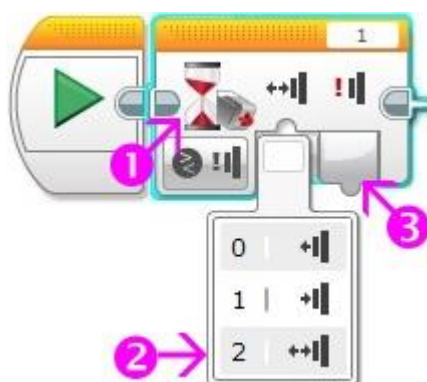


Рисунок 7 – Элемент программы

Итак: при такой настройке блока ожидания выполнение нашей программы будет остановлено до нажатия-отпускания кнопки датчика касания. Только после "Щелчка" выполнение будет передано следующему программному блоку. Установим после блока ожидания один программный блок "Рулевое управление", загрузим программу в робота и убедимся в правильности её выполнения! (Рисунок 8)



Рисунок 8 – Вид программы

Задача 3.3: необходимо написать программу, останавливающую робота, столкнувшегося с препятствием.

Из датчика касания давайте соберем небольшой бампер, который будет нам сигнализировать о том, что наш робот столкнулся с препятствием. Ниже приведены подробные инструкции для сборки (рисунок 9), как из домашней, так и из образовательной версии конструктора Lego mindstorms EV3. Можете поэкспериментировать и придумать собственный вариант конструкции.

Lego mindstorms EV3 home

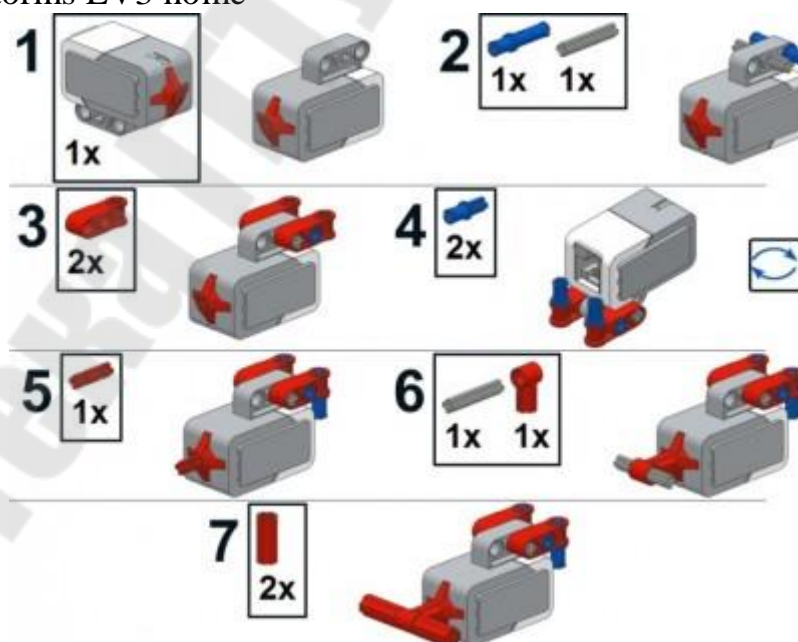


Рисунок 9 – Сборка бампера

Получившийся элемент закрепим на передней балке нашего робота (рисунок 10) и соединим датчик касания с портом "1" модуля EV3.

Lego mindstorms EV3 Home

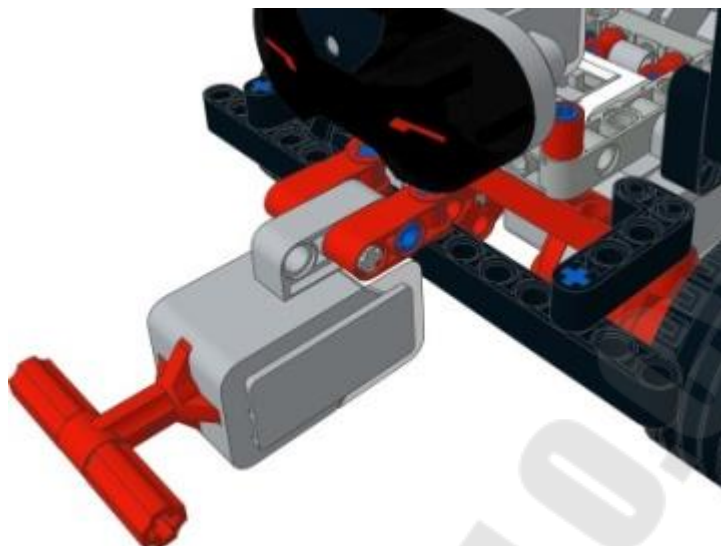


Рисунок 10 – Размещение бампера на роботе

Конструкция готова! Приступим к созданию программы. По условию задачи: робот должен двигаться вперед, пока не наткнется на препятствие. В этом случае датчик касания будет нажат! Для решения снова воспользуемся программным блоком "Ожидание".

Решение:

Начать прямолинейное движение вперед (Рисунок 11 поз. 1).

Ждать, пока датчик касания не будет нажат (Рисунок 11 поз. 2).

Прекратить движение вперед (Рисунок 11 поз. 3).



Рисунок 11 – Решение задачи объезда препятствий

Для решения следующей задачи нам понадобится программный блок "Цикл" Оранжевой палитры.

Задача 3.4: необходимо написать программу, заставляющую робота двигаться вперед, при наезде на препятствие - отъезжать назад, поворачивать вправо на 90 градусов и продолжать движение вперед до следующего препятствия.

Подсказка: напишите и протестируйте программу движения - отъезда - поворота, а затем поместите эти блоки внутри программного блока "Цикл"

Алгоритм решения задачи 3.4:

1. Включаем моторы для прямолинейного движения вперед;

2. Ожидаем нажатия датчика касания;
3. Выключаем моторы;
4. Отъезжаем немного назад;
5. Расчитываем значения параметра для поворота робота вправо на 90 градусов (диаметр колес робота равен 56 мм (образовательная версия конструктора));
6. Поворачиваем вправо на 90 градусов;
7. Пункты 1 - 6 повторяем в бесконечном цикле.

4 .Оформить отчет:

Титульник, цель, ход работы: задание, блок-схема решения с комментариями; выводы.

Лабораторная работа № 2

«Работа с датчиками EV3» (часть 1 2 часа)

1 Цель работы

Получить навыки работы с модулем Lego Mindstorms EV3, научиться подключать различные датчики, устройства вывода, сигнальные и исполнительные устройства.

2 Теоретические сведения

2.

2.1 Изучить документацию по контроллеру Lego EV3

2.2 Собрать на практическом занятии робота-тележку

2.3 Изучить принцип работы тактильного датчика (кнопки) и датчика цвета.

На этом занятии продолжаем знакомство с датчиками набора Lego mindstorms EV3. На очереди - датчик цвета, очень важный и полезный датчик! В большинстве конструкций он является тем, чем у человека являются глаза. Поэтому изучению датчика цвета мы посвятим два последовательных урока, но в дальнейшем курсе еще вернемся к его изучению и использованию.

Датчик цвета может работать в трех различных режимах:

- в режиме "Цвет" датчик может определить цвет поднесенного к нему предмета;
- в режиме "Яркость отраженного света" датчик направляет световой луч на близкорасположенный предмет и по отраженному пучку определяет яркость предмета;
- в режиме "Яркость внешнего освещения" датчик может определить - насколько ярко освещено пространство вокруг.



Рисунок 1 – Внешний вид датчика цвета

Задание: Датчик цвета. Режим "Цвет"

В режиме "Цвет" датчик цвета достаточно точно умеет определять семь базовых цветов предметов, находящихся от него на расстоянии примерно в 1 см. Это следующие цвета: "черный"=1, "синий"=2, "зеленый"=3, "желтый"=4, "красный"=5, "белый"=6 и "коричневый"=7. Если предмет удален от датчика или некорректно определяется цвет предмета - датчик информирует об этом состоянием "Без цвета"=0.

Задача 3,1: необходимо написать программу, называющую цвета предметов, подносимых к датчику цвета.

Если вы собирали своего тренировочного робота по инструкции, то у вас датчик цвета уже размещен внутри робота и направлен вниз (рисунок 2). Потребуется приложить некоторые усилия, может быть даже слегка разобрать-собрать нашу конструкцию, чтобы подключить кабелем датчик цвета, например к порту "2" модуля EV3. Для отладки программы нам также понадобится несколько цветных предметов: это могут быть кирпичики конструктора Lego, полоски цветной бумаги или цветные кубики. Для лучшего результата следует взять цвета, максимально приближенные к основным, но датчик довольно неплохо справляется с распознаванием подходящих оттенков. Чтобы не снимать датчик цвета и не крепить его в другом месте, во время выполнения программы можно держать робота перевернутым вверх колесами.

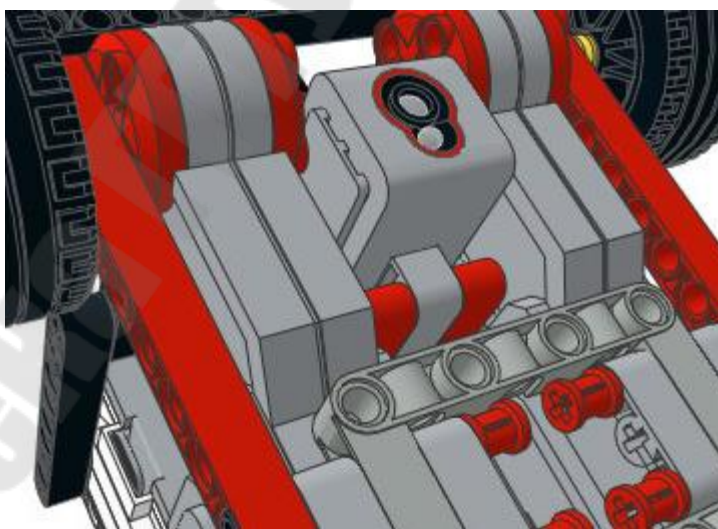


Рисунок 2 – Расположение датчика цвета

Оранжевая палитра, программный блок "Переключатель"

В решении Задачи 3.1 нам поможет программный блок "Переключатель" Оранжевой палитры. Этот блок в зависимости от настроек выбирает для выполнения программные блоки, расположенные в одном из своих контейнеров. Рассмотрим настройку этого блока в режиме работы с датчиком цвета.

Создадим новую программу "lesson-5-9", установим в программе блок "Переключатель", выберем режим "Датчик цвета" - "Измерение" - "Цвет" (Рисунок 3). В отличие от программного блока "Ожидание", программный блок "Переключатель" не ждет, пока наступит определенное событие, а проверяет текущее состояние и выполняет программные блоки, находящиеся в контейнере, сопоставленном текущему состоянию.

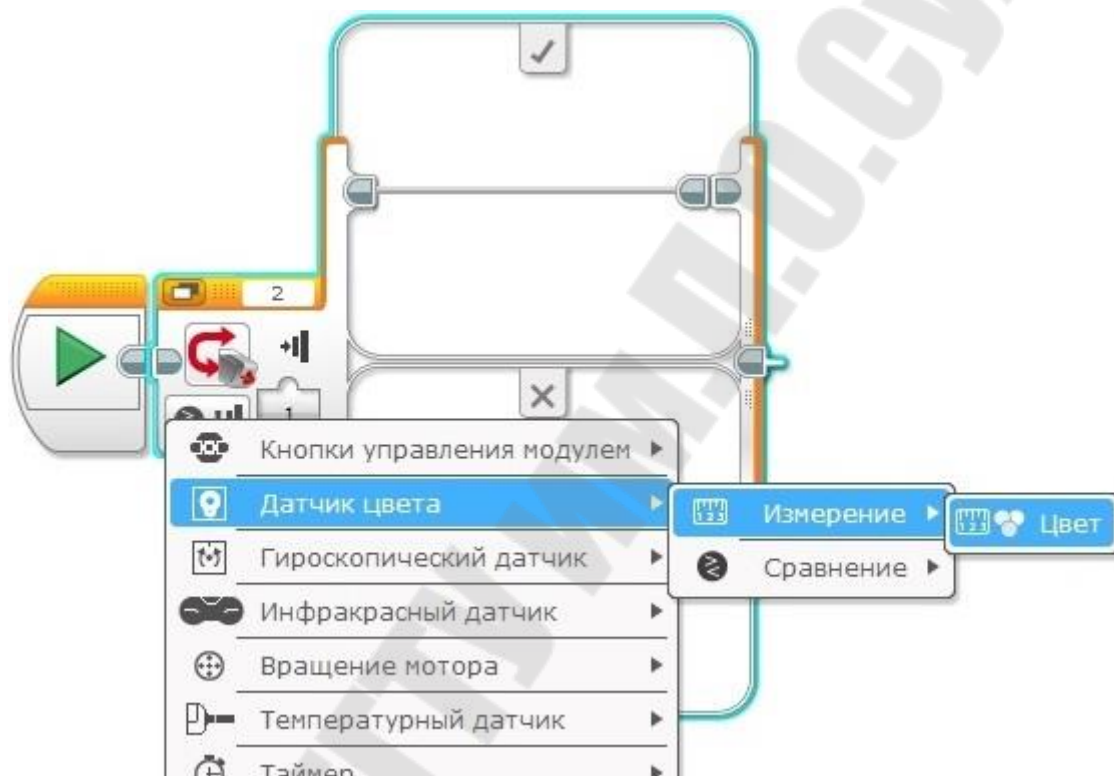


Рисунок 3 – Программное подключение датчика цвета

Рассмотрим подробнее настройки программного блока "Переключатель":

- выбранный режим устанавливает изображение датчика цвета в блоке (Рисунок 4 поз. 1),
- порт, к которому подключен датчик, отображается в соответствующем поле блока (Рисунок 4 поз. 2),
- в настройках каждого программного контейнера выбирается значение, в соответствии с которым будут выполняться программные блоки, вложенные в этот контейнер (Рисунок 4 поз. 3),
- один из контейнеров должен быть объявленным "Вариантом по умолчанию" - в случае, если значению, полученному от датчика, не соответствует ни один контейнер, то выполняется контейнер, объявленный "Вариантом по умолчанию" (Рисунок 4 поз. 4),
- Кнопка "+" добавляет программный контейнер в блоке "Переключатель" (Рисунок 4 поз. 5),
- Программный блок "Переключатель" может автоматически растягиваться, чтобы вместить все блоки, помещаемые внутрь. С

помощью меток, помеченных красными стрелками, можно самому изменять размеры блока (Рисунок4).

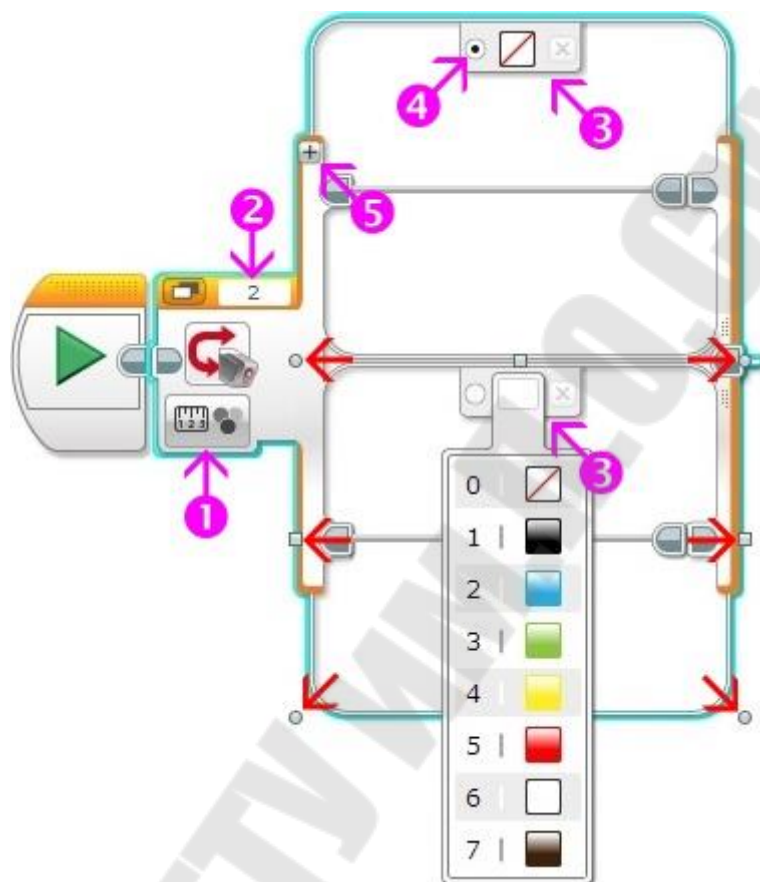


Рисунок 4 – Настройки режимов измерения/сравнения

Продолжим формирование программного блока "Переключатель":

- создадим необходимое количество контейнеров, соответствующее количеству цветов для распознавания + вариант "Без цвета",
- в настройках контейнеров установим распознаваемые цвета,
- вариантом по умолчанию выберем вариант "Без цвета",
- в каждый контейнер кроме варианта "Без цвета" (этот контейнер останется пустым) поместим программный блок "Звук" зеленой палитры.
- каждому цвету сопоставим соответствующий звуковой файл.

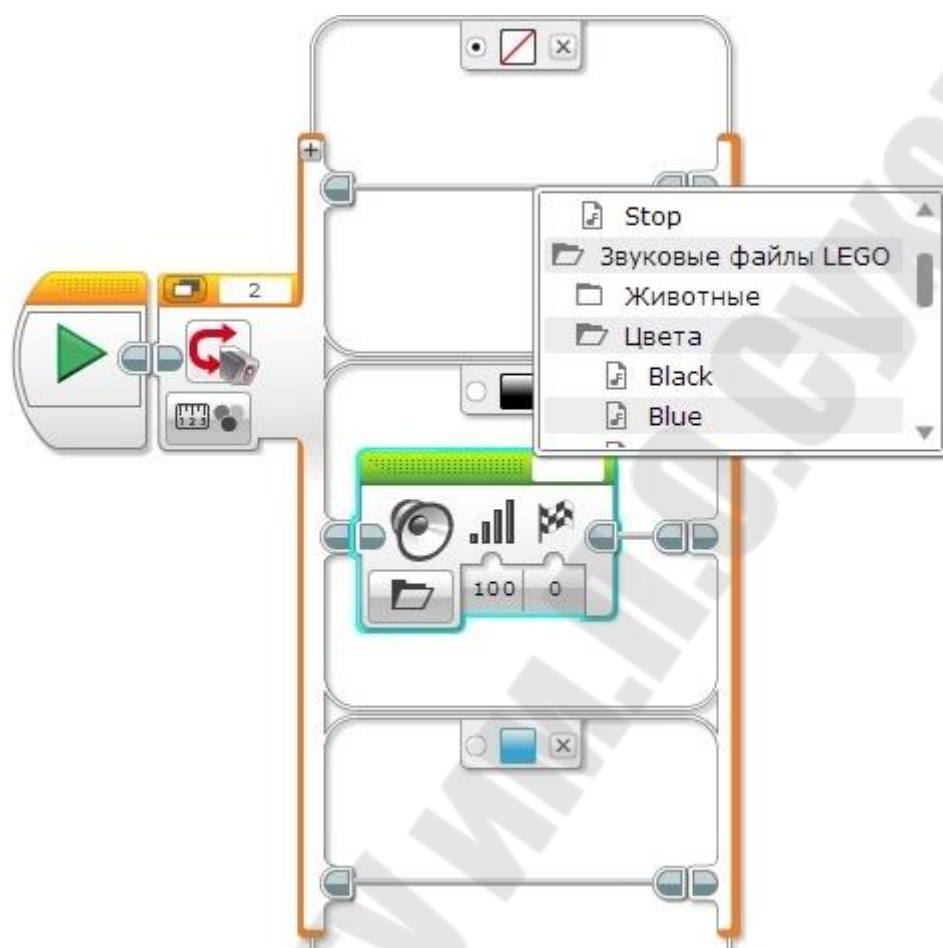


Рисунок 5 – Добавление звукового сопровождения

Наш программный блок "Переключатель" значительно увеличился в размерах. Специальная кнопка (Рисунок 6 поз. 1) позволяет переключить режим отображения блока на экране на "Вид с вкладками". Изменим размеры блока для комфортного визуального отображения.

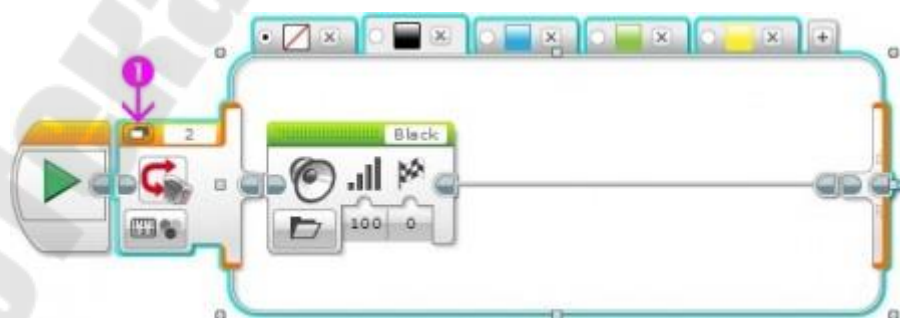


Рисунок 6 – Масштабирование внешнего вида программы

Осталось вставить наш настроенный программный блок "Переключатель" внутрь программного блока "Цикл" Оранжевой палитры. Программа готова. Загрузим её в робота и протестируем работу. (Рисунок 7)

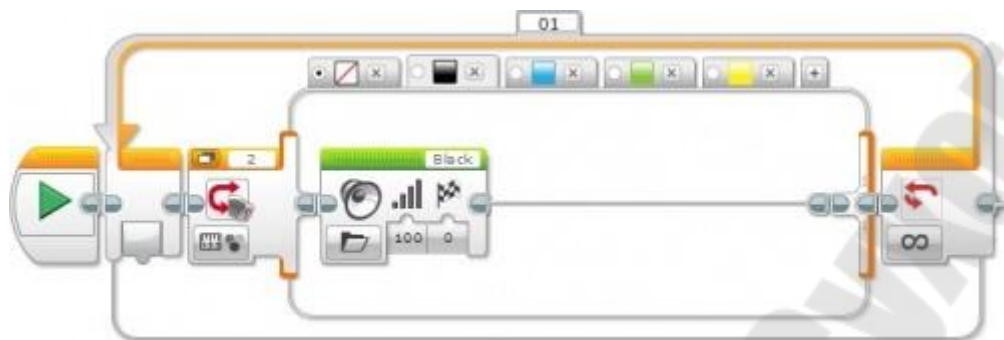


Рисунок 7 – Решение поставленной задачи

Оранжевая палитра, программный блок "Прерывание цикла"

Добавим в нашу программу движение. Сделаем следующее поле для выполнения задания:

Возьмем белый лист бумаги формата A4 или A3;

Нанесем на него последовательно, на равном расстоянии несколько цветных полос. Полосы можно наклеить из цветной бумаги, цветной изолянт или нарисовать и закрасить;

можете также загрузить подготовленное изображение и распечатать его на цветном принтере;

Последнюю полосу сделаем черного цвета (Рисунок 8).

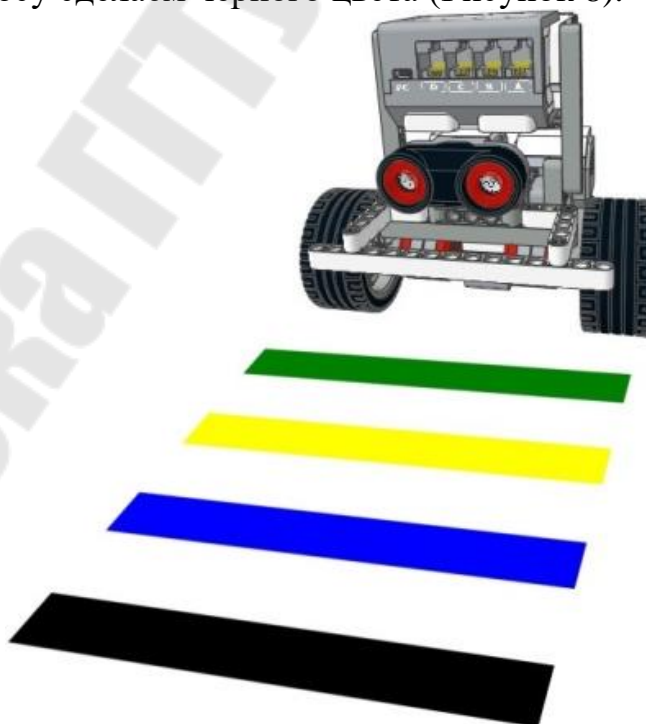


Рисунок 8 – Создание палитры для следующей задачи

Задача 3,2: необходимо написать программу прямолинейного движения робота, называющего цвета полос, над которыми он проезжает. При достижении черной полосы робот проговаривает "Stop" и останавливается.

За основу решения данной задачи возьмем программу, решающую предыдущую задачу. При решении задачи 3.2 потребуется прервать выполнение цикла. Этой цели служит программный блок "Прерывание цикла" Оранжевой палитры. С помощью данного блока можно организовать выход из цикла, заданного параметром "Имя прерывания" (Рисунок 9 поз. 1).

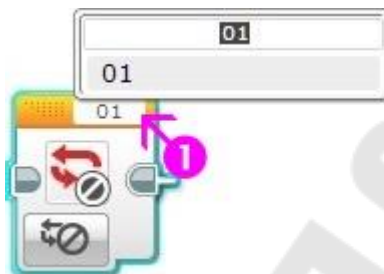


Рисунок 9 – Прерывание цикла

Попробуйте решить Задачу 3.2 самостоятельно, не подглядывая в решение.

Алгоритм решения задачи 3.2.

Внутри цикла перед программным блоком "Переключатель" добавим программный блок "Рулевое управление", тем самым заставим нашего робота двигаться. Во время движения робот будет проверять текущее состояние датчика цвета и произносить название цвета. Если полосы будут широкими, а робот будет двигаться медленно, то, возможно, он станет произносить название цвета более одного раза, так как проверка цвета будет происходить неоднократно. Если такое положение дел вас не устроит - увеличьте скорость робота, чтобы он быстрее проезжал цветные полосы.

В соответствии с условием задачи нам надо изменить поведение контейнера программного блока "Переключатель" для черного цвета.

В программном блоке "Звук" изменим звуковой файл "Black" на "Stop".

Добавим в контейнер программный блок, выключающий моторы.

Нам требуется прервать выполнение программного блока "Цикл", чтобы завершить выполнение программы. Для этого поместим в контейнер программный блок "Прерывание цикла" Оранжевой палитры. У данного программного блока существует только одна настройка - название прерываемого цикла. В сложной программе со множеством циклов важно правильно устанавливать эту настройку, чтобы остановить выполнение нужного цикла. В нашей программе за программным блоком "Цикл" отсутствуют другие программные блоки, поэтому программа завершится.

4 .Оформить отчет:

Титульник, цель, ход работы: задание, блок-схема решения с комментариями; выводы.

Лабораторная работа № 3

Программирование роботов из стандартных наборов на Arduino.

(2 часа)

1 Цель работы

Получить навыки работы с готовыми робототехническими средствами на базе Arduino.

2 Теоретические сведения:

2.1 Изучить на практическом занятии техническую документацию по роботу в соответствие с вариантом (выдается преподавателем: pop-bot, sparky, hexy).

2.2 Ознакомиться с соответствующей средой программирования робота.

3 Задание

3.1 Подключиться к роботу в соответствующей IDE/ проверить заявленный в техдокументации функционал.

3.2 Получить в соответствие с вариантом задание от преподавателя по модернизации функций (озвучивание действий, изменение функционала пульта управления, добавление автоматических реакций на посторонние предметы в контролируемых областях).

3.3 Написать соответствующие скетчи и проверить их работоспособность.

3.4 Оформить и защитить отчет.

Лабораторная работа № 4

Программирование роботов из стандартных наборов на Lego EV3.

(2 часа)

1 Цель работы

Получить навыки работы с готовыми робототехническими средствами на базе Lego EV3.

2 Теоретические сведения:

2.1 Изучить на практическом занятии техническую документацию по роботу в соответствие с вариантом и собрать его (выдается преподавателем: pop-bot, sparky, hexu).

2.2 Ознакомиться с соответствующей средой программирования робота.

3 Задание

3.1 Подключиться к роботу в соответствующей IDE/ проверить заявленный в техдокументации функционал.

3.2 Получить в соответствие с вариантом задание от преподавателя по модернизации функций (озвучивание действий, изменение функционала пульта управления, добавление автоматических реакций на посторонние предметы в контролируемых областях).

3.3 Написать соответствующие блок-схемы программ и проверить их работоспособность.

3.4 Продемонстрировать обновленный функционал преподавателю.

3.4 Оформить и защитить отчет.

Лабораторная работа № 6

Программирование движения роботов по заданной траектории.

(1 час)

1 Цель работы:

Изучить основные принципы организации движения робототехнических средств.

2 Теоретические сведения:

2.1 Подготовить робота (EV3 или Arduino)

2.2 Организация движения робота на примере «тележки на EV3»

Запускаем среду программирования Lego mindstorms EV3, загружаем наш проект lessons.ev3 и добавляем в проект новую программу - lesson-3-4. Добавлять новую программу в проект мы научились с вами на предыдущем уроке.

Красная палитра – операции с данными

Программные блоки, необходимые для выполнения различных операций над числовыми, логическими или текстовыми данными, сосредоточены в красной палитре среды программирования Lego mindstorms EV3. Красная палитра содержит 10 программных блоков. В отличие от зеленой палитры - с программными блоками красной палитры мы будем знакомиться постепенно, по мере продвижения по курсу программирования и возникновения необходимости в новых программных конструкциях.

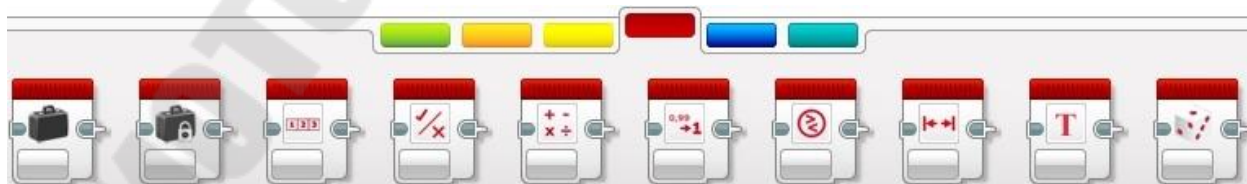


Рисунок1 – Палитра «Операции с данными»

Числовые значения. Блок "Константа", блок "Переменная"

Среда программирования Lego mindstorms EV3 позволяет нам обрабатывать в своих программах пять различных типов данных: "Текст", "Числовое значение", "Логическое значение", "Числовой массив", "Логический

массив". В сегодняшнем уроке мы научимся оперировать с числовыми данными. Тип данных "Числовое значение" позволяет нам выполнять различные математические операции над числами. Числа в программе могут быть как положительными, так и отрицательными, быть целыми значениями или содержать десятичную дробь. Примеры: -15; 3,145; 8; -247,34.

Перед тем, как начать обрабатывать различные типы данных в наших программах, нам надо научиться их создавать и хранить. Для этих целей среда программирования Lego mindstorms EV3 предоставляет два вида программных блоков: "Переменная" и "Константа". Эти блоки позволяют создать в памяти робота специальные ячейки, позволяющие записывать, извлекать и редактировать различные типы данных. Программный блок "Константа" (Рисунок 2) позволяет создавать ячейку памяти для хранения одного из пяти типов данных (Рисунок 2 поз. 1). Требуемое значение записывается в ячейку на этапе создания программы (Рисунок 2 поз. 2) и остается неизменным во время выполнения всей программы. Для получения значения, записанного в блок "Константа" используется "Вывод" (Рисунок 2 поз. 3).

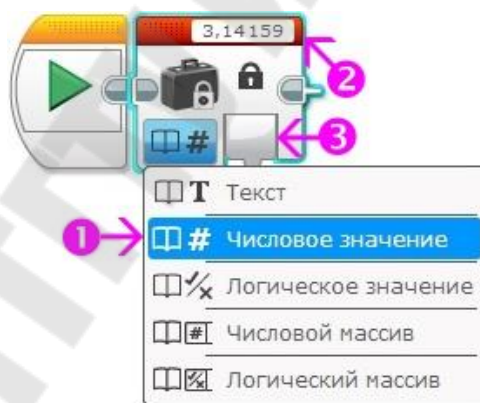


Рисунок 2 – Программный блок «Константа»

В отличие от программного блока "Константа" - в блоке "Переменная" присутствуют два режима "Считывание" и "Записать" (Рисунок 3 поз. 1). Перед первым использованием необходимо задать имя переменной, выбрав параметр блока "Добавить переменную" (Рисунок 3 поз. 2). Имя переменной может содержать только заглавные и строчные буквы латинского алфавита, цифры, а также символы _ и -. Задать значение переменной можно, записав или передав число в параметр "Значение" (Рисунок 3 поз. 3).

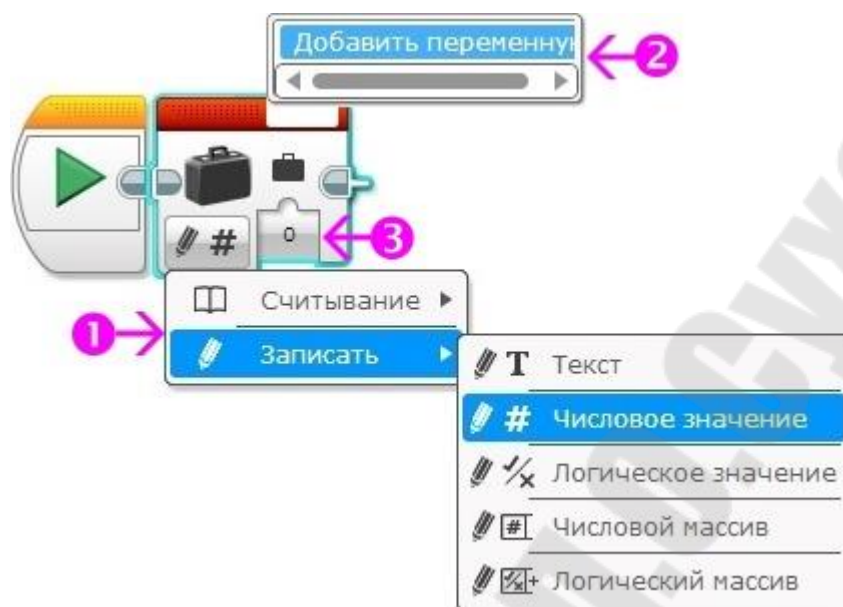


Рисунок 3 – Программный блок «Переменная»

Блок математика, блок округление

Для выполнения математических вычислений служит программный блок "Математика". Он позволяет выполнить выбранную математическую операцию (Рисунок 4 поз. 1) над двумя числами, заданными параметрами "a" и "b". В режимах "Абсолютная величина" и "Квадратный корень" для вычисления доступен только один параметр "a".

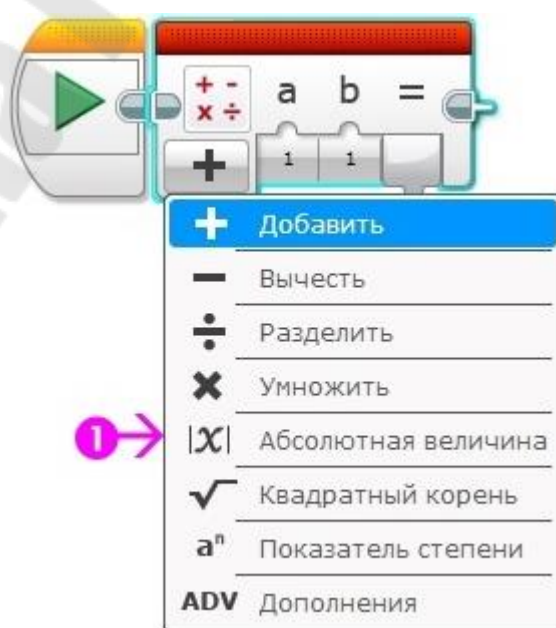


Рисунок 4 – Программный блок «Математика»

Отдельно следует остановиться на режиме "Дополнения". В этом режиме количество параметров для расчета увеличивается до четырех: "a", "b", "c" и "d". В параметр "Уравнение" (Рисунок 5 поз. 1) можно вписать любую произвольную формулу, производящую вычисления с этими параметрами.

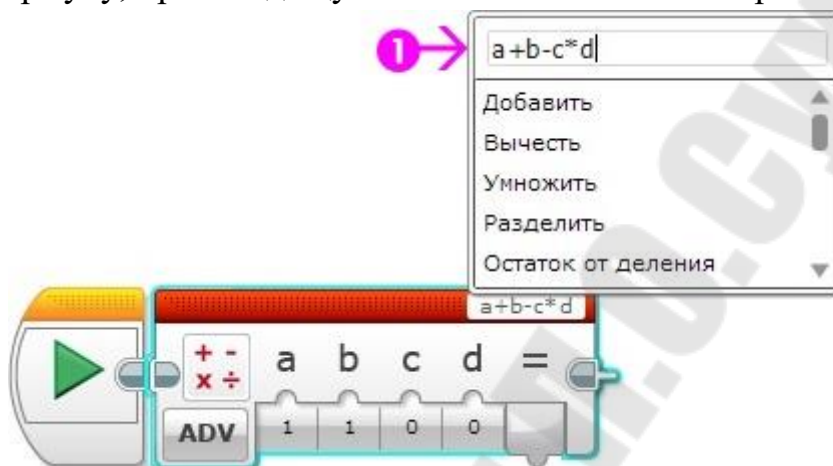


Рисунок 5 – Программный блок «Математика» режим «Дополнение»

Иногда возникает необходимость произвести округление результата вычисления. Например: при отладке программы, можно выводить на экран модуля EV3 округленные промежуточные расчеты, чтобы легче было визуально контролировать ход выполнения программы. Для этого предназначен программный блок "Округление" (Рисунок 6). Режимы "До ближайшего", "Округлить к большему" и "Округлить к меньшему" производят округление до целого значения. В режиме "Отбросить дробную часть" можно задать количество остающихся знаков дробной части после запятой.

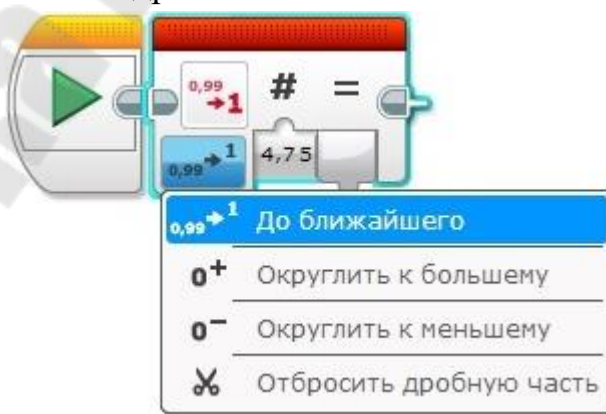


Рисунок 6 – Округление

3 Задание

Задача 3.1 : необходимо написать программу прямолинейного движения для проезда роботом расстояния в 1 метр.

Решение:

За один полный оборот мотора робот проезжает расстояние, равное длине окружности колеса. Это расстояние можно найти, умножив число Π ($\approx 3,14159$) на диаметр колеса. Диаметр колеса из образовательного набора Lego mindstorms EV3 равен 56 мм, а - из домашнего набора Lego mindstorms EV3 равен 43,2 мм. Если переведем расстояние в 1 метр в миллиметры (1000 мм) и разделим на расстояние, которое робот проходит за один оборот мотора, то узнаем: сколько оборотов мотора необходимо для проезда всего заданного расстояния.



Рисунок 7 – Схема расчета расстояния

Приступим к созданию программы:

Используя программный блок "Константа", заведем в программу постоянное число Π , равное примерно 3,14159.



Рисунок 8 – Загрузка числа « Π »

Используя программный блок "Переменная", создадим в программе переменную D и занесем в нее значение диаметра колеса в зависимости от используемого конструктора (если вы использовали другие колеса, то самостоятельно измерьте диаметр и внесите значение в программный блок).



Рисунок 9 – Фиксация диаметра колеса

Используя программный блок "Математика", умножим значение блока "Константа" на значение переменной D. Для передачи значения из переменной

В программный блок "Математика" используем второй программный блок "Переменная" в режиме "Считывание"! (Для передачи значений между программными блоками используются шины данных. Чтобы установить шину данных, необходимо "потянуть" выходной параметр одного программного блока и "присоединить" его к входному параметру другого программного блока)

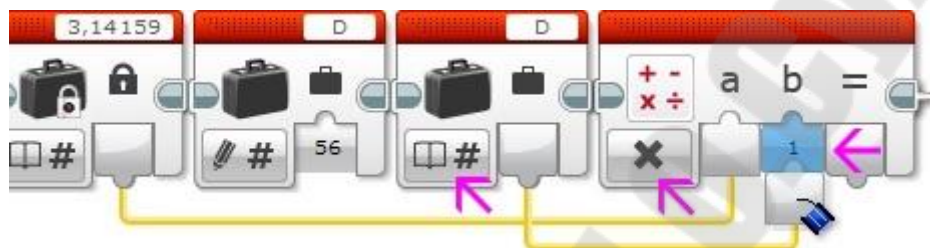


Рисунок 10 – Применение «шины данных»

Используя программный блок "Математика", разделим значение пути (1000 мм) на значение, полученное в шаге 3.

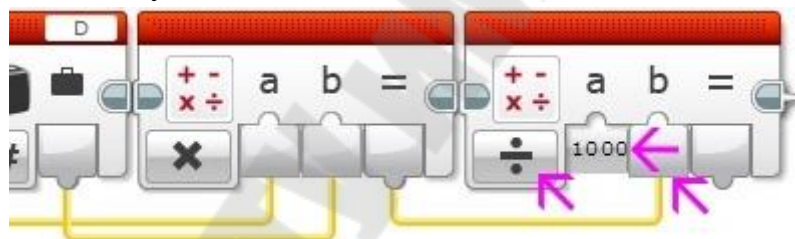


Рисунок 11 – Математические вычисления

Полученное в предыдущем шаге значение. округлив до двух знаков после запятой, выведем на экран модуля EV3.

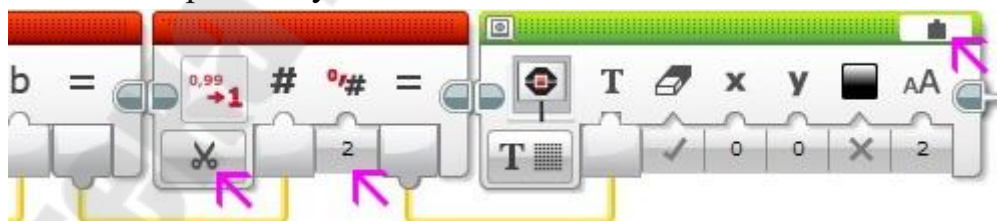


Рисунок 12 – Округление и вывод результатов

Полученное в шаге значение подадим в параметр "Обороты" блока "Рулевое управление".

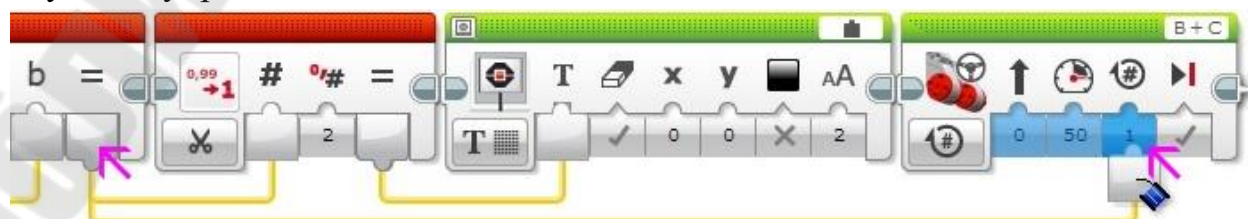


Рисунок 13 – Назначение оборотов

Загрузим полученную программу в нашего робота. Поставим робота на ровную свободную площадку и запустим программу. Измерив расстояние, пройденное роботом, убедимся в правильности нашей программы.

Задача 3.2: необходимо написать программу, рассчитывающую значение параметра "Градусы" для разворота нашего робота.

Данная задача имеет сходство с предыдущей - нам только требуется найти расстояние, которое должны проехать колеса нашего робота. Для того, чтобы наш робот развернулся на 180 градусов - необходимо, чтобы правое и левое колеса, проехав определенный путь по окружности, поменялись местами. Как видим из Рисунок 8 - каждое колесо при этом проедет ровно половину окружности с диаметром, равным расстоянию между центрами колес (красная линия на Рисунок 8). Подходящей линейкой померяем расстояние между центрами колес. Для робота, собранного по инструкции small-robot-45544, это расстояние равно 120 мм. Следовательно, умножив это значение на число Пи (3,14159) и разделив на 2, мы найдем расстояние, которое должно проехать каждое из колес нашего робота. Как найти соответствующее этому расстоянию число оборотов мотора - мы разобрали в Задаче 4 данного урока. Для того, чтобы перевести полученное число оборотов в градусы - вспомним соотношение: 1 оборот мотора = 360 градусов. Следовательно, если мы, воспользовавшись программным блоком "Математика", умножим полученное значение оборотов на 360 и подадим результат в параметр "Градусы" программного блока "Независимое управление моторами" (Урок №2 Рисунок7 поз. 2), то решим требуемую задачу.

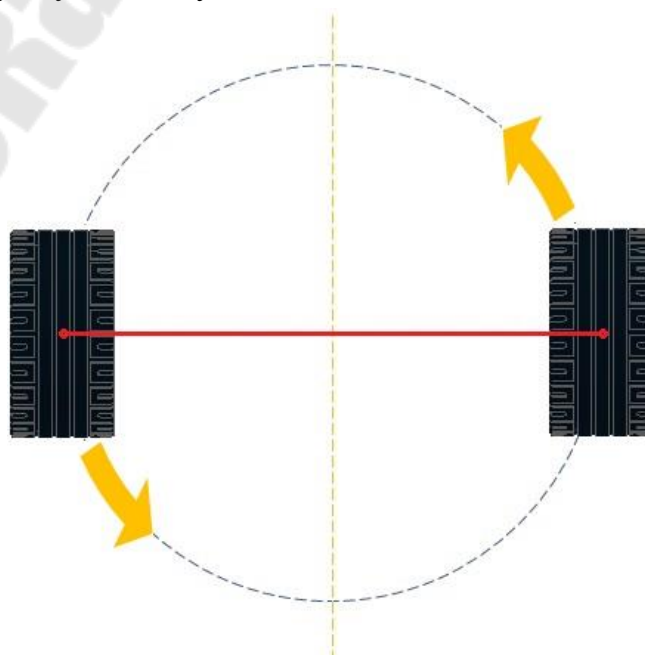


Рисунок 14 – Схема вычисления расстояний для разворота

4 .Оформить отчет:

Титульник, цель, ход работы: задание, блок-схема решения с комментариями; выводы.

Лабораторная работа № 6

Работа с периферийных устройств с одноплатными ЭВМ.

(2 часа)

1 Цель работы:

Изучить основные принципы подключения датчиков к одноплатным компьютерам, научиться считывать с них информацию и выводить ее на экран.

2 Теоретические сведения:

2.1 На практическом занятии в соответствии с вариантом подготовить образ и развернуть операционную систему на соответствующем одноплатном компьютере (Raspberry PI, Orange PI, Banana PI, Asus ThinkerBoard)

2.2 Изучить в технической документации к одноплатному компьютеру назначение выходов порта GPIO.

2. 3Подключение датчика ds18b20 к RPi3

2.3.1 Подключим датчик температуры к GPIO Raspberry Pi 3. Его нужно установить так (рисунок 1):

- напряжение – 3,3 Вольта;
- заземление;
- порт GPIO4.

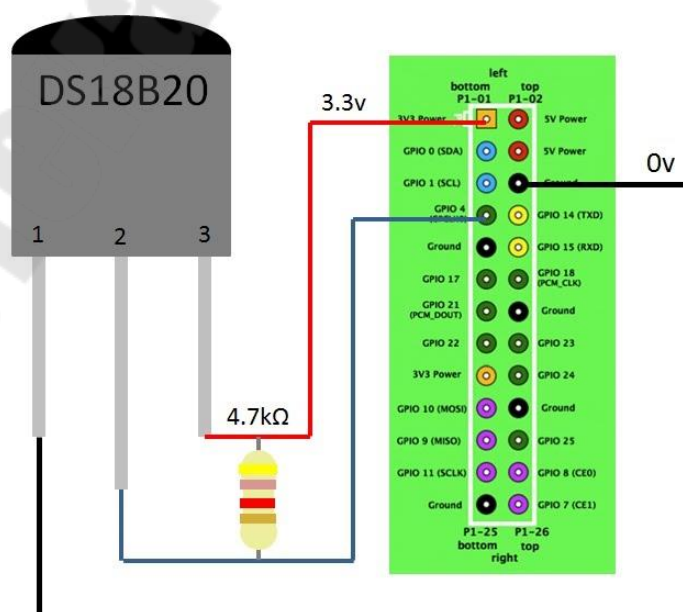


Рисунок 1 – Подключение датчика температуры к GPIO

2.3.2 Установка необходимых модули ядра.

Из-под `sudo` (для этого можно указать в консоли команду `sudo -i` и ввести пароль) требуется выполнить команды:

`modprobe w1-gpio` и `modprobe w1-therm`.

Первый модуль нужен, чтобы на порту GPIO4 активировался протокол 1-wire.

Второй же необходим, чтобы система могла непосредственно считывать температурные показатели с соответствующей шины.

После перезагрузки модули снова будут неактивны, поэтому их придется опять загружать. Но исправить это очень легко. Нужно всего лишь вписать их названия в файл `/etc/modules`. Для этого из-под `sudo` следует набрать команду `nano /etc/modules` и добавить в конец файла `w1-gpio` и `w1-therm` (каждый с новой строки), а затем сохранить изменения.

2.3.3 Проверить работоспособность датчика и, если все нормально, писать код для работы с ним или использовать готовое решение.

На Raspberry Pi 3 датчики пишут данные в файлы. Конкретно `ds18b20` записывает их каталог со своим серийным номером, который, в свою очередь, располагается в директории `/sys/bus/w1/devices`. Он появляется после подключения, но может не сразу, а спустя несколько секунд.

Название соответствующего каталога примерно такое: `00-000000000a0b`. Поэтому после подключения зайдите и проверьте, есть ли там что-то похожее. Следует отметить, что если подключены несколько датчиков, то придется узнавать серийный номер каждого из них опытным путем. Это возможно сделать, например, прогревом и снятием показателей или поочередным подключением.

Чтобы проверить работоспособность датчика нужно в консоли вписать следующую команду: **`cat СЕРИЙНЫЙ_НОМЕР/w1_slave`**.

После нажатия на `Enter` выведется сообщение.

В первой строке будет присутствовать контрольная сумма в шестнадцатеричной системе счисления и CRC. Если значение `crc` равно последнему числу, то выводится YES, а если нет – NO. В первом случае датчик сообщает, что он правильно определил температуру, а во втором – нет.

Вторая строка – это как раз данные температуры. То, что указывается в шестнадцатеричной системе полностью соответствует значению выше. То, что левее – это его перевод в десятичную форму. Интересно то, что значение `t` (температура) указывается в 1000 раз больше, чем есть на самом деле. Поэтому если после знака "=" указывается, например, 26398, это означает 26,3 градуса Цельсия. Данный факт следует учитывать при написании кода.

3. Задание

3.1 Изучить документацию по подключению 2х строчного дисплея.

3.2 Подключить дисплей к выходам GPIO одноплатного компьютера.

3.3 Инициализировать дисплей

3.4 Вывести показания датчика на экран.

Продemonстрировать результаты работы преподавателю. Подготовить отчет.

Лабораторная работа № 7

Подключение приводов к одноплатным ЭВМ.

(2 часа)

Цель работы:

Изучить основные принципы подключения исполнительных механизмов к одноплатным компьютерам, научиться управлять ими.

2 Теоретические сведения:

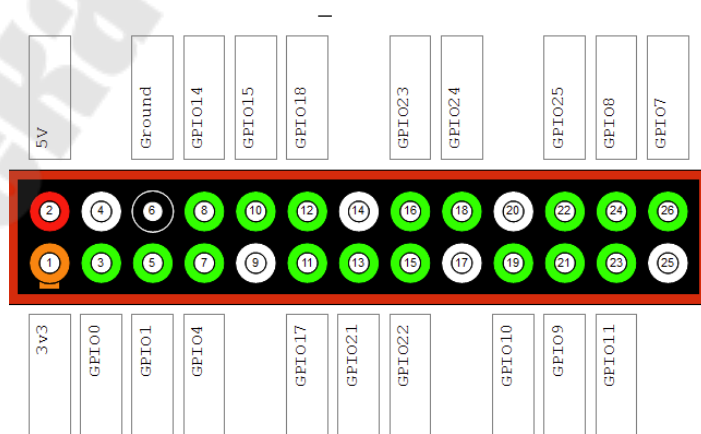
2.1 На практическом занятии в соответствии с вариантом подготовить образ и развернуть операционную систему на соответствующем одноплатном компьютере (Raspberry PI, Orange PI, Banana PI, Asus ThinkerBoard)

2.2 Изучить в технической документации к одноплатному компьютеру назначение выходов порта GPIO.

2.3 Подключение сервопривода к одноплатному ПК

От сервопривода идет шлейф из трех проводов:

- *красный* – *питание* – подключается к контакту 5V (пин 2) или непосредственно к источнику питания;
- *коричневый или черный* – земля GND (пин 6 Raspberry Pi);
- *желтый или белый* – сигнал; подключим к пину 7.



– Рисунок 1 – разъем GPIO

Спецификации нашего сервопривода:

Модель: *TowerPro SG90*

Рабочее напряжение: *4.8 DC*

Скорость: *60 градусов за 0.1 с*

Крутящий момент: *1.6 кг см*

Вес: *9 г*

Диапазон ширины импульса: *0.5 мс – 2.5 мс*

Данный сервопривод не может вращаться на 360 градусов. Он имеет нейтральную позицию (*Neutral*) и границы в ноль и 180 градусов. Для установления серво в нейтральную позицию, необходимо подать высокий сигнал длительностью 1.5 мс, в 0 градусов – 0.5 мс, в 180 градусов – 2.5 мс.

Рассмотрим следующую программу:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
p = GPIO.PWM(7,50)
p.start(7.5)
try:
    while True:
        p.ChangeDutyCycle(7.5)
        time.sleep(1)
        p.ChangeDutyCycle(12.5)
        time.sleep(1)
        p.ChangeDutyCycle(2.5)
        time.sleep(1)
except KeyboardInterrupt:
    p.stop()
    GPIO.cleanup()
```

В данной программе мы настраиваем седьмой пин для работы широтно-импульсной модуляции с частотой 50 Герц: `GPIO.PWM(7,50)`

Теперь подавая на серво импульсы разной длины мы можем изменять его положение

3 Задание

Заставить сервопривод вращаться по алгоритму 3 оборота по часовой стрелке => одни- против => два по часовой.

Продемонстрировать работу преподавателю, подготовить отчет.