Описание лабораторного оборудования

Микроконтроллер STAMP

В Stamp используются младшие модели Microchip PIC (конкретно в BS2 - Microchip PIC16C57c), которые не реализуют прерывания и не содержат каких-нибудь аппаратных средств, работающих одновременно с основной программой. Специальные характеристики АЛУ (арифметико-логическое устройство) и линий ввода-вывода реализуются программно, что дает гораздо большую гибкость, чем специализированный микроконтроллер. Но это также приводит к тому, что Stamp выполняет некоторые специальные функции медленнее, чем микроконтроллеры, в которых они реализованы аппаратно.

BASIC Stamp2 имеет архитектуру показанную на рис.1. Программный счетчик имеет стек на четыре элемента для хранения адреса возврата из подпрограмм, что позволяет иметь до трех вложенных подпрограмм.

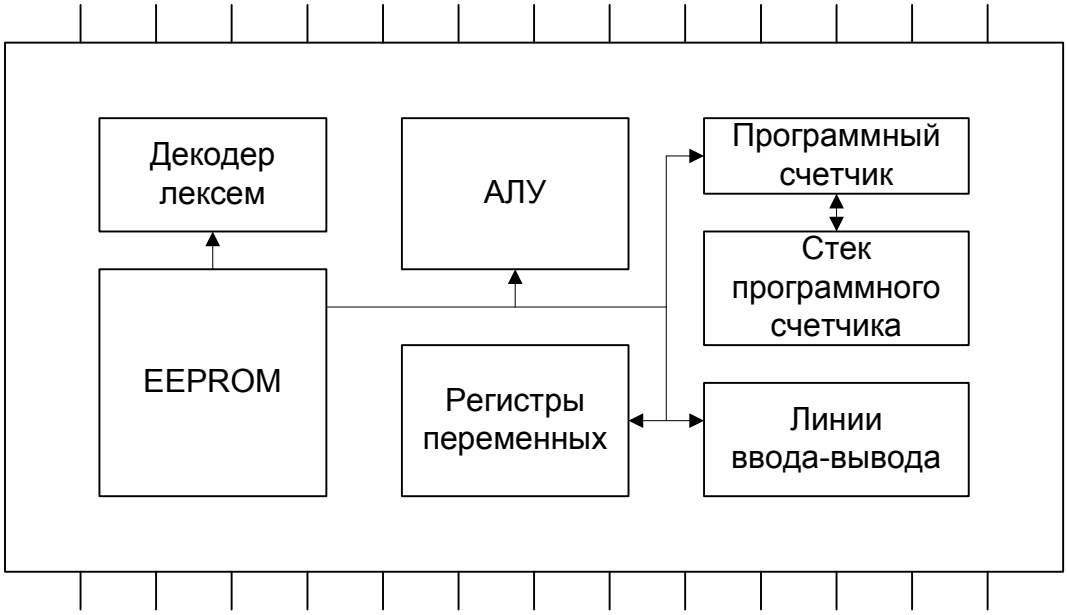


Рис. 1. Структурная схема контроллера BS2

Программные «лексемы»

Одной важной особенностью, которую надо иметь в виду при работе со Stamp - это то, что он выполняет не обычные команды, а «лексемы», поступающие из памяти EEPROM. Лексемы - это маленькие команды, которые формируются из исходного кода. Обычно они имеют длину два или три байта, которые последовательно выбираются из EEPROM. Последовательная передача данных - основная причина низкой производительности Stamp по сравнению со встраиваемыми микроконтроллерами.

Как только лексема загружена, она выполняется при помощи вызова соответствующей функции. Этими функциями могут быть арифметические операции, пересылки данных, проверки условий, операции управления выводами или последовательный обмен с системным/пользовательским интерфейсом управляющего компьютера. Из-за различий в длине и возможном времени выполнения лексем очень трудно определить реальное время выполнения программы. Фактически приходится определять это время эмпирически, измеряя время работы приложения.

Память EEPROM с последовательной выборкой

Когда программа написана и откомпилирована в лексемы, ее надо загрузить с управляющего компьютера и контроллер Stamp, который, в свою очередь, загружает ее во внутреннюю память EEPROM с последовательной выборкой. Эта память служит не только для хранения программ - ее можно также использовать для хранения переменных.

Преимущество такого использования памяти EEPROM состоит в том, что данные сохранятся в Stamp даже после отключения от него питания.

В контроллерах Stamp размещение программы начинается с конца EEPROM, а адреса располагаются последовательно в порядке убывания, так как обращение к памяти EEPROM реализуется с декрементом адреса (см. рис. 2). Запись и чтение можно проводит в любом месте EEPROM.

Это означает, что надо соблюдать осторожность, чтобы преобразованный в лексемы исходный код не был записан на место уже существующего.

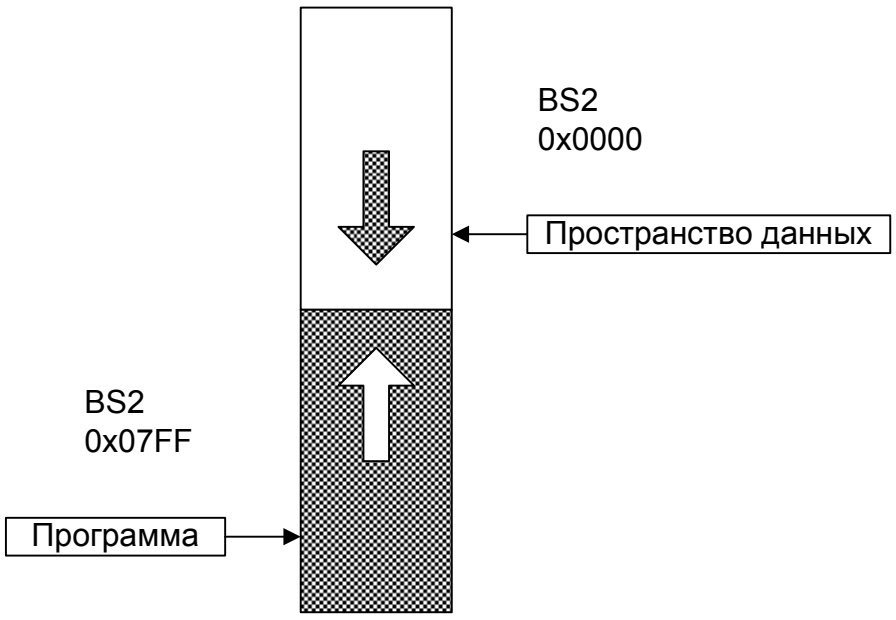


Рис. 2. Организация памяти EEPROM с последовательной выборкой в Basic Stamp

Интерфейсы программирования

Контроллер Stamp имеет двунаправленный последовательный интерфейс, который используется как для программирования EEPROM с последовательной выборкой, так и для передачи отладочной информации для управляющего компьютера во время выполнения программы. Контроллер Stamp II использует последовательный порт для приема и передачи данных при программировании (см. рис. 3).

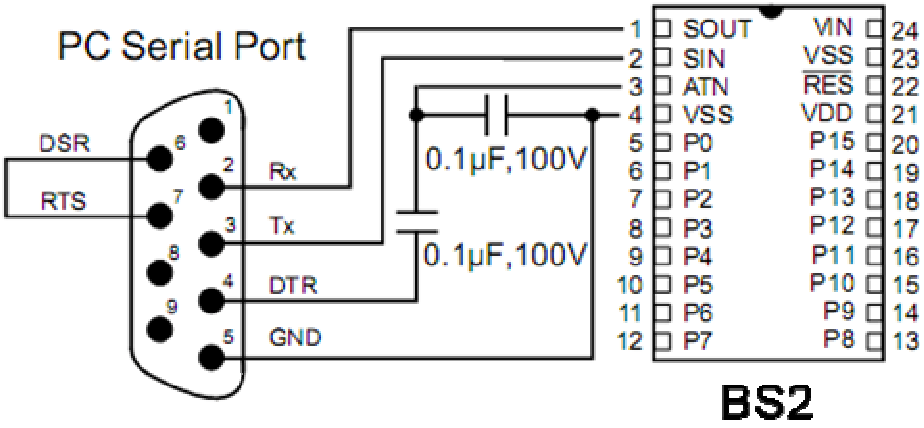


Рис. 2.1 Интерфейс программирования контроллера BS2

Среда разработки Basic Stamp Development System

Для написания программ Parallax предоставляет специальную среду разработки Basic Stamp Development System (рис. 4) для операционной системы Windows.

Среда Basic Stamp Development System представляет собой привычный Windows редактор, состоящий из основного поля где выводится текст программы и интегрированного “проводника” по файловой системе компьютера. Среда разработки позволяет открывать одновременно до шестнадцати программ и переключаться между ними нажатием на закладку с именем программы; умеет выделять команды, конструкции и специфические выражения выбранного языка программирования. С помощью нее также происходит непосредственное программирование контроллера Stamp. Для этого необходимо подключить макетную плату к порту компьютера (USB или COM) и подать на нее питание. Следует отметить, что если среда разработки не обнаружит контроллер, то попытки запуска программ будут завершаться ошибкой.

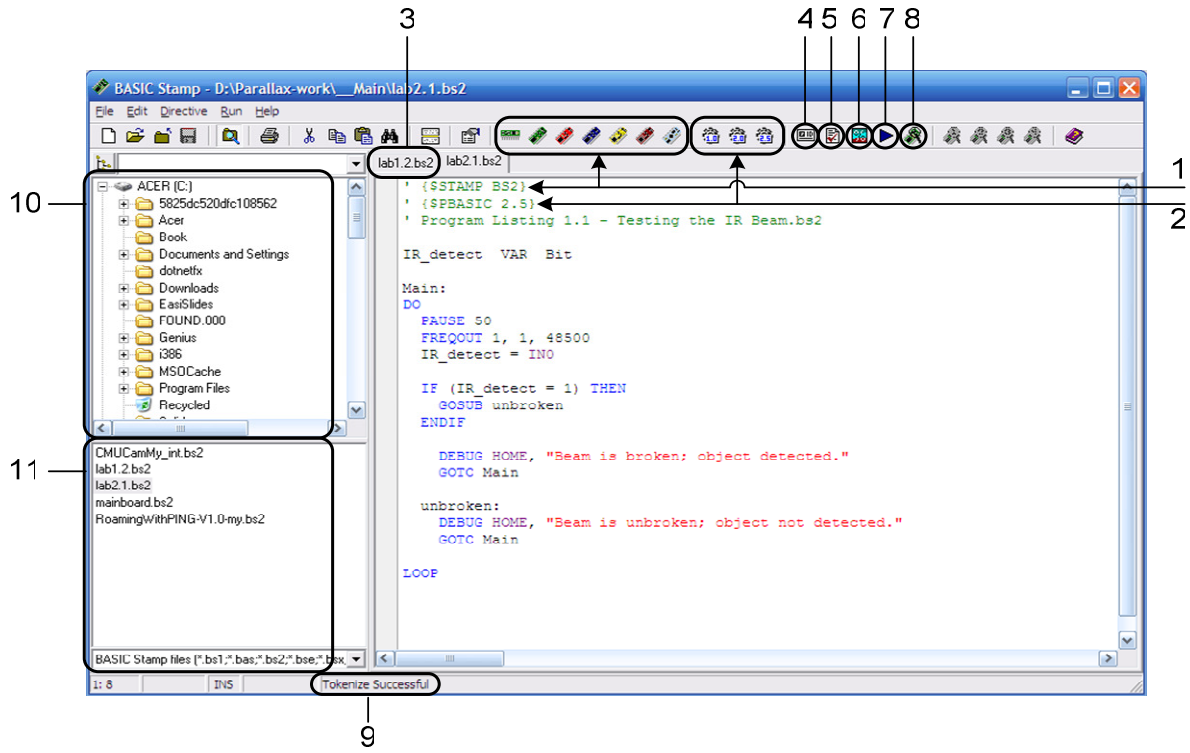


Рис. 2.2. Графический интерфейс среды Basic Stamp Development System

1. Директивы, сигнализирующие компилятору, для какого микроконтроллера написана программа;
2. Директивы, сигнализирующие компилятору, на какой версии языка написана программа;
3. Вкладки, позволяют в одном окне программы держать открытыми несколько проектов;
4. `Identify`. Идентификация микропроцессора, подключенного к компьютеру (обычно выполняется автоматически при загрузке программы);
5. `Syntax Check`. Запуск проверки синтаксиса написанной программы;
6. `Memory Map`. Выводит карту свободной и занятой памяти в графическом виде;
7. `Run`. Загрузка написанной программы в память микроконтроллера;
8. `New Debug`. Открытие нового окна терминала отладки;
9. `Tokenize Successful`. Это сообщение означает, что проверка синтаксиса прошла успешно – написанную программу можно смело загружать в микроконтроллер;
10. Дерево папок компьютера;
11. Проекты, находящиеся в выбранной папке.

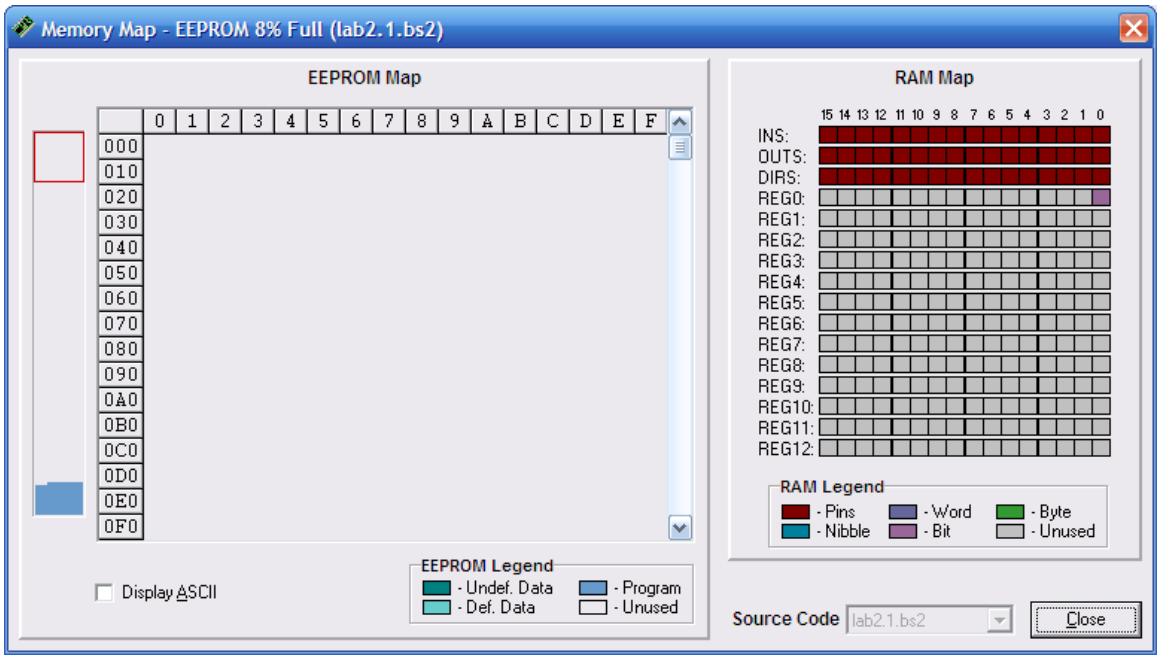


Рис. 2.3. Окно Memory Map

Организация кода

Организацию кода рекомендуется формировать в следующем порядке:

1. Объявление директив;
2. Декларация контактов микроконтроллера;
3. Объявление переменных;
4. Объявление констант;
5. Инициализация переменных (присвоение начальных значений);
6. Основной текст программы;
7. Подпрограммы (процедуры).

Синтаксис языка программирования PBasic 2.5

Директивы компилятора

Директивы компилятора являются обязательными, они указывают среде программирования контроллер, версию языка PBasic и порт к которому подключен робот. Указывать порт необходимо только в том случае, если к компьютеру одновременно подключено более одного контроллера.

Синтаксис директив:

Тип контроллера: `'{$STAMP <версия>}`

Версия языка программирования: `'{$PBASIC <версия>}`

Порт: `'{$PORT COM<номер>}`

Декларация контактов микроконтроллера

Декларация контактов необходима для присвоение понятных человеку названий контактам микроконтроллера, к которым подключены те или иные датчики и устройства.

Например: к контакту №12 микроконтроллера подключен правый сервопривод. Можно во всех местах программы, где это необходимо, указывать просто цифру 12, но, для повышения читабельности кода программы и облегчения понимая вашего кода другими людьми, рекомендуется назвать этот контакт микроконтроллера понятным словосочетанием «RightServo».

Синтаксис:

```
<название_контакта>    PIN      <номер_контакта>
```

Пример: `RightServo  PIN  12`

Объявление переменных и констант

Прежде чем использовать в программе переменную, её нужно объявить. Т.е. указать компилятору ее имя и тип. Это необходимо для того, чтобы компилятор выделил под переменную место в памяти и привязал его к переменной.

В таблице приведены типы переменных PBasic версии 2.5 и их размер.

Таблица 1

№	Тип	Размер, бит	Возможные значения
1.	BIT	1	0 – 1
2.	NIB	4	0 – 15
3.	BYTE	8	0 – 255
4.	WORD	16	0 – 65535 -32768 – 32767

В языке PBasic наряду с одномерными переменными, существуют многомерные (массивы). Объявление массива происходит точно так же, как и одномерной переменной, только при указании типа переменной в скобках указывается его размерность. Обращение к элементу массива происходит указанием имени массива и номером нужного элемента в скобках. Нумерация элементов массива начинается с нуля. Следовательно, если был объявлен массив с 10-ю элементами, то нумероваться они будут от 0 до 9.

Иногда возникает потребность в «переменных», которые не должны изменяться в ходе программы, их так же называют константами. Язык PBasic позволяет определять константы. Объявление константы отличается от объявления обычной переменной. Вместо ключевого слова VAR, необходимо использовать CON, а вместо типа, нужно указывать значение константы.

Следует отметить, что в языке PBasic, как и в большинстве других языков программирования, также присутствуют предопределенные константы. Здесь предопределенные константы содержат в себе значения на портах ввода/вывода контроллера. (табл. 2).

Таблица 2

Тип	**Word**	**Byte**	**Nibble**	**Bit**
Ввод	INS	INL, INH	INA, INB, INC, IND	IN0 – IN15
Вывод	OUTS	OUTL, OUTH	OUTA, OUTB, OUTC, OUTD	OUT0 – OUT15
Направление	DIRS	DIRL, DIRH	DIRA, DIRB, DIRC, DIRD	DIR0 – DIR15

INS – значение на вводе (доступны только на чтение).

OUTS – значения на выводе.

DIRS – управление направление ввод/вывод на порту.

Переменные могут быть разбиты на переменные меньших размеров путем использования «модификаторов переменных», которые являются простой формой структур данных. Используются следующие модификаторы переменных:

HIGHBYTE	Старший байт слова
BYTE1	Старший байт слова
LOWBYTE	Младший байт слова
BYTE0	Младший байт слова
HIGHNIB	Старший полубайт слова или байта
NIB3	Старший полубайт слона
NIBI	Старший полубайт байта или полубайт I слова
LOWNIB	Младший полубайт слова или байта
NTB2	Полубайт 2 слова
NIB0	Младший полубайт слова или байта
HIGHBIT	Старший бит слова, байта или полубайта
LOWBIT	Младший бит слона, байта или полубайта
BIT0	Бит0 слова, байта или полубайта
BIT1	Бит1 слова, байта или полубайта
BIT7	Бит7 слова или старший бит байта

В языке PBasic переменные могут быть только числовые, строковых переменных нет.

Имя переменной это некоторая комбинация латинских букв и цифр, обязательно начинающаяся с буквы.

Синтаксис объявления переменных и констант:

```
<имя_переменной>   VAR       <тип_переменной>
<имя_константы>    CON       <значение_константы>
```

Инициализация переменных

При необходимости присвоения какого-либо начального значения переменным, это можно сделать до основного кода программы.

Например: `firstAngle = 0` (обнуление переменной «`firstAngle`»).

Текст программы

Программа - это конечная последовательность команд и операций иногда оформленная в виде некоторых конструкций. Рассмотрим, что представляют и какими бывают команды, операции и конструкции в языке PBasic 2.5.

Команды в PBasic.

Команды в языках программирования - это зарезервированные слова, которые подразумевают под собой некоторую, определенную для каждой команды, последовательность действий.

В данном пособии рассмотрены только основные команды, которые будут использоваться в лабораторных работах. С остальными командами можно ознакомиться через справочную систему среды Basic Stamp Development System или с помощью справочного пособия «BASIC Stamp Syntax and Reference Manual».

Таблица 2.1

Команды для управления ходом программы и отладки	
PAUSE [T]	Останавливает выполнение команды на время, указанное в качестве параметра (в миллисекундах).
SLEEP [T]	Останавливает выполнение команды на время, указанное в качестве параметра (в миллисекундах). В отличии от PAUSE во время команды SLEEP уменьшается энергопотребление.
END	Конец программы, контроллер переходит в режим ожидания. Уменьшается энергопотребление.
DEBUG [MSG]	Отсылает информацию на компьютер. Приходящую от робота информацию можно посмотреть с помощью Терминала отладки (Debug Terminal's Recive).
Команды для управления портами ввода/вывода контроллера	
HIGH [#]	Устанавливает на порту, заданном в качестве параметра [#], значение напряжения соответствующее единице.
LOW [#]	Устанавливает на порту, заданном в качестве параметра [#], значение напряжения соответствующее нулю.
PULSEIN [#],[S],[V]	Измеряет на заданном порту [#] длительность импульса и записывает результат в переменную [V] (в микросекундах), для получения корректного результата, переменная должна

	быть типа Word. Параметр [S] задает, какие импульсы считать: из “единиц” (1) или “нулей” (0).
PULSOUT [#],[N]	Генерирует заданное количество [N] 2х-микросекундных импульсов на указанном порту [#].
FREQOUT [#],[T],[F]	Генерирует на указанном порту [#] синусоиду заданной длительности [T] (в миллисекундах) и заданной частоты [F] (в герцах).
SERIN `[#IP]{/[#OP]}, [S], [Vs]`	Асинхронный прием данных через COM-порт. [#IP] –порт, работающий на ввод, [#OP] – порт, работающий на вывод (необязательный параметр), [S] – скорость передачи данных, [Vs] – переменная(ые), куда будут записаны полученные данные.
SEROUT `[#OP]{/[#IP]}, [S], [Vs]`	Асинхронная отправка данных через COM-порт. [#IP] – порт, работающий на вывод, [#OP] – порт, работающий на ввод (необязательный параметр), [S] – скорость передачи данных, [Vs] – передаваемые данные.

Синтаксис основного кода программы:

```
Main:
     DO
     .
     . (код программы)
     .
     LOOP
```

Операции в PBasic

Операции – это действия, которые может совершать процессор контроллера с данными, хранящимися в памяти. Контроллер Basic Stamp 2 работает только с числовыми переменными, поэтому его процессор выполняет только арифметические и логические операции. Кроме того, контроллер работает только с целочисленными переменными, это следует помнить при выполнении операции деления. В таблице 4 приведено соответствие символов и операций.

Таблица 2.2

Операция	Описание
+	Арифметическое сложение.
–	Арифметическое вычитание.
*	Арифметическое умножение.

/	Арифметическое деление. Результат – целое число.
=	Логическое “равно”, либо операция присвоения.
<	Логическое “меньше”.
<=	Логическое “меньше, либо равно”.
>	Логическое “больше”.
>=	Логическое “больше, либо равно”.
<>	Логическое “неравно”.
AND	Логическое “и”
OR	Логическое “или”

Операторные выражения

Операторное выражение - это формула, расположенная справа от знака равенства («=») в выражениях присваивания или справа от оператора «IF» в условных выражениях. Хотя это похоже на формат, используемый в других языках, есть одна важная особенность, которая может вызвать проблемы при разработке программ для Stamp. Эта особенность заключается в том, как вычисляются выражения, преобразуемые, в конечном счете, в лексемы. В контроллерах Stamp - эти выражения вычисляются слева направо независимо от приоритета выполнения операций (при выполнении арифметических операции и порядке приоритета умножение выполняется до сложения или вычитания, но не раньше сравнения). При создании программного кода для этого контроллера следует обратить особое внимание на то, чтобы операции выполнялись в правильном порядке. Например, если мы должны выполнить следующее выражение присваивания:

A = A * B + C * D ,

то во многих языках оно вычисляется в следующем порядке:

1. A * B
2. C * D
3. Результат 1 + Результат 2
4. Сохранить Результат 3 в «A»

Так как BASIC Stamp выполняет все операции слева направо, это данное выражение присваивания будет вычисляться следующим образом:

1. A * B
2. (A * B) + C
3. (A * B) + C) * D
4. Сохранить Результат 3 в «A»

Если в BASIC Stamp 2 две операции умножения заключены в круглые скобки:

A = (A * B) + (C * D)

то выражение будет вычисляться с «нормальным» порядком выполнения операций. Контроллер BASIC Stamp 2 может обрабатывать до 8-ми уровней круглых скобок, что позволяет нам писать программы, реализующие необходимый порядок выполнения операций.

Еще одни дополнительная сложность в работе с BASIC Stamp 2 - это реализация им унарных операций (операций, имеющих только один входной параметр), таких как NOT, и бинарных операций, таких как AND, прежде чем будут выполнены бинарные арифметические операции, такие как «+», «*» и «/». Самый простой способ обеспечить необходимый порядок выполнения операций в BASIC Stamp 2 это помнить, что операции, которые задаются словами, имеют более высокий приоритет, чем те, которые используют символы арифметических выражений.

При разработке приложений рекомендуется после любого сложного операторного выражения использовать команду «DEBUG» для распечатки результата выражения до тех пор, пока вы хорошо не познакомитесь со способом, которым PBASIC для Stamp вычисляет выражение.

Все вышесказанное касается также операций отношения, которые являются частью операторных выражений.

Унарные операторы

В таблице 2.3 приведены некоторые из унарных операторов.

Таблица 2.3

SQR	Квадратный корень беззнаковой величины
-	Преобразование 16-битового числа в дополнительный код
~	Инверсия битов 16-битового числа
ABS	Абсолютное значение 16-битового числа
DCD	Декодирование 4-битового числа в 16-битовое
NCD	Приоритетное кодирование 16-битового числа
SIN	Синус 8-битового числа
COS	Косинус 8-битового числа

Выполнение умножения с помощью операторов «*» и «**» - производится умножение 16-битных чисел, которое дает 32-битный результат. Выполнение одной «звездочки», позволяет получить младшее слово или 16 бит результата. Операция «двойной звездочки» дает значение старших 16 бит результата. Это может быть показано на примере:

```
A = $4
B = $1234     'Некоторые  величины для умножения
C = $10
```

```
Test = A * B    ' Test = $48 DO
Test = A ** B   ' Test = $0000
Test = B * C    ' Test = $2340
Test = B ** C   ' Test = $0001
```

Операция деления ("/") следует за умножением, но отличается от неё тем, что делимое и делитель могут иметь максимум 16 бит.

Операторы «Min» и «Max», которые не используют никаких специальных символов и просто представляются как метки, выделяют, соответственно, минимальное или максимальное из представленных значений, например:

```
Test = 5 max 7          ' Test = 7
```

PBASIC не воспринимает отрицательные числа, они будут обрабатываться как числа, у которых старший бит установлен в 1, что заставит PBASIC думать, что в данный момент проверяется большое положительное число. В этом случае результат операции не будет соответствовать ожидаемому.

Двумя последними унарными операторами являются тригонометрические функции «синус» и «косинус» - «SIN» и «COS», соответственно. Эти функции выдают 8-битное число от -127 до 127, которое является значением соответствующем тригонометрической величины для окружности с 256 точками, имеющей радиус 127 единиц (рис. 2.4).

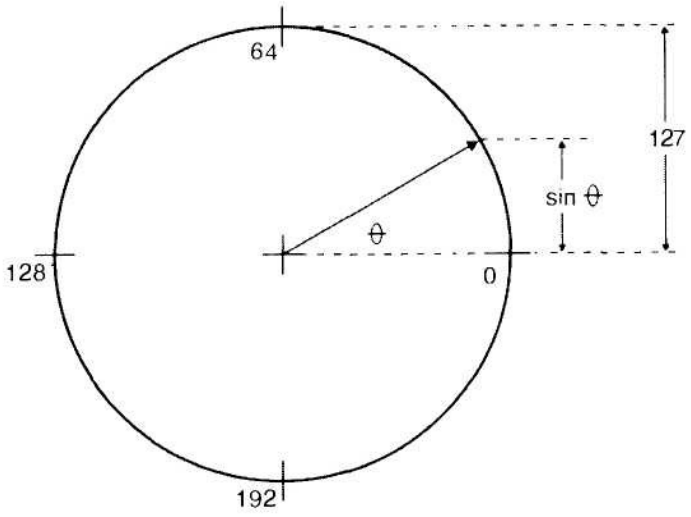


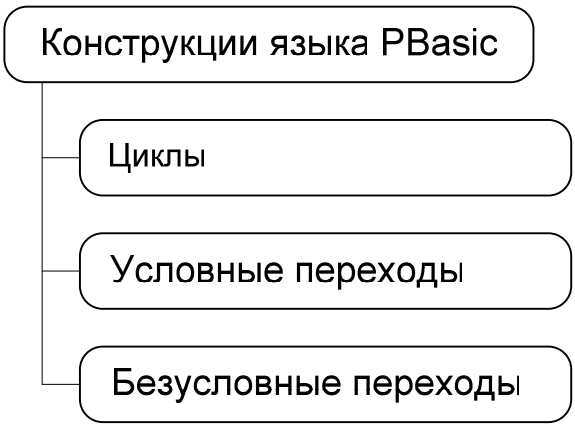
Рис. 2.4 Окружность для вычисления синуса и косинуса в Stamp

Каждый квадрант окружности разбит на 64 условных «градуса» (вместо 90 обычных угловых градусов). Выходная величина (реальное тригонометрическое значение) может быть вычислена как 1/127 от полученного результата.

Реализация операции «дробного умножения» («*/») является очень интересным свойством языка PBASIC для BS2. При умножении двух 16-

битных чисел результат может содержать до 32-х значениях разрядов. Оператор «*/» выделяет два средних байта результата, игнорируя старшин и младший байты. Это делает ее очень полезным для нахождения дробных значений.

Конструкции языка PBasic



Циклы

Цикл – это последовательность одинаковых действий. Цикл может выполняться заранее известное число раз, бесконечно, либо прекращать выполняться при выполнении какого-либо условия. Для организации циклов в языке PBasic существует две конструкции.

Первая конструкция **FOR … NEXT** нужна для организации цикла с известным числом итераций (повторений).

```
FOR <счетчик> = <нач. знач.> TO <кон. знач.> STEP <шаг>
   …
NEXT
```

`<счетчик>` – переменная, считающая количество сделанных повторений.

`<нач. знач.>` – начальное значение счетчика

`<кон. знач.>` – конечное значение счетчика

`<шаг>` – значение, на которое увеличивается счетчик при одном проходе.

Выполнение конструкции `FOR … NEXT` прекращается, когда значение счетчика будет больше указанного конечного значения. Внутри цикла можно применять любые команды, операции и конструкции языка PBasic.

Вторая конструкция для организации циклов **DO … LOOP**. Она позволяет организовывать циклы с предусловием (когда решение о проходе шага цикла принимается перед шагом), с постусловием (решение о проходе следующего шага принимается после прохода предыдущего шага) и бесконечный цикл. Принципиальная разница между циклами с предусловием и постусловием заключается в том, что цикл с постусловием выполнится хотя бы один раз, потому как решение о следующем шаге

принимается после прохода шага, а цикл с предусловием может не выполнится ни разу.

```
DO {WHILE | UNTIL <условие>}
 …
LOOP {WHILE | UNTIL <условие>}
```

Ключевые слова WHILE и UNTIL определяют условие выхода/продолжения цикла.

Если используется слово WHILE, то при выполнение условия [условие] цикл будет продолжать выполняться, если [условие] не выполняется, то цикл будет прекращен.

При использовании слова UNTIL будет происходить обратное, при выполнении – выход, при невыполнении – продолжение цикла.

Ключевые слова WHILE и UNTIL должны использоваться в конструкции только один раз, либо после DO, либо после LOOP. Если ключевое слово стоит после DO, цикл будет с предусловием, если после LOOP – цикл с постусловием. Если ключевых слов не указано, цикл будет бесконечным.

Условные переходы

Условные переходы нужны для выполнения действий, в зависимости от определенных условий. Т.е. при выполнении одного условия, выполняются одни действия, при выполнении других условий – другие.

Конструкция **IF … THEN**

```
IF <условие_1> THEN
 … <действие_1> …
ELSEIF <условие_2> THEN
 … <действие_2> …
…
ELSE
 … <действие_N> …
ENDIF
```

Конструкция начинается с ключевого слова IF и заканчивается ключевым словом ENDIF. При выполнении <условие_1> после слова IF, выполняются действия заключенные между ключевым словом THEN и ELSEIF, ELSE или ENDIF. При не выполнении условия, действия пропускаются. Если дальше стоит ключевое слово ELSEIF, то проверяется условие, стоящее после этого слова, при его выполнении так же выполняются действия, записанные после THEN, и так же пропускаются, если условие не выполнено. Действия, записанные после ключевого слова ELSE, выполняются при невыполнении условия, записанного после IF или ELSEIF, стоящего перед ELSE.

Конструкция **BRANCH**

BRANCH <переменная>, [<метка_0>, <метка_1>, … , <метка_N>]

Конструкция начинается с ключевого слова BRANCH, затем идет переменная и после, в квадратных скобках, перечень меток (про то, что такое метки и их синтаксис рассказано при описании безусловных переходов). Переход происходит на метку, порядковый номер которой в списке соответствует значению переменной, записанной после BRANCH. Нумерация меток начинается с нуля. Если значение переменной превышает количество меток, то перехода не происходит, и выполняются действия, записанные после конструкции.

Конструкция **SELECT … CASE**.

Конструкция является смесью двух предыдущих.

```
SELECT <переменная>
 CASE {<условие>} <значение>
      … <действия> …
ENDSELECT
```

После ключевого слова SELECT записывается переменная, после ключевых слов CASE записывается условие (оператор сравнения) и какое либо значение, для сравнения с ним переменной. Если оператор сравнения – знак равенства, то его можно опустить. Так же возможно записывать условие попадания переменной в заданные промежутки (промежутки записываются через запятую). Синтаксис записи промежутков:

<нач. значение> TO <кон. значение>

При попадании значения переменной под условие, стоящее после CASE, выполняются действия, записанные между этим CASE и следующим (либо ENDSELECT). После выполнения, начинают выполняться действия, идущие после конструкции, т.е. записанные после ключевого слова ENDSELECT.

Если вместо условия со значением стоит слово ELSE, то действия выполнятся, если не выполнено ни одно из предыдущих условий.

Безусловные переходы

Безусловные переходы нужны для переходов внутри программы. Переходы возможны только в места, обозначенные специальным образом, к так называемым меткам.

Метка представляет собой строку из одного слова, заканчивающегося двоеточием. Именем метки будет являться это слово, но без двоеточия. Переход к метке осуществляется с помощью оператора **GOTO**, после которого записано имя метки.

В следующем примере:

```
…
a = 10 + 5
GOTO Jump
a = a – 9
Jump:
DEBUG DEC a
…
```

строка «`a = a – 9`» выполнена не будет, потому что после оператора `GOTO` произойдет переход к метке Jump.

С помощью безусловных переходов можно выделять часто повторяющиеся действия в программе и не писать их каждый раз, а просто ставить переход к ним. Однако в таком способе есть один большой недостаток. Так как вызов выделенной части программы (подпрограммы) производится в разных местах программы, то необходимо знать, куда следует возвращаться после ее выполнения. Эту проблему можно решить несколькими способами, например, вставить в конец подпрограммы ветвления и в зависимости от выполнения некоторых условий возвращаться в нужные места, однако данный вариант не удобен при большом числе вызовов подпрограммы. Как раз для таких случаев существует конструкция **`GOSUB`**. Синтаксис `GOSUB` такой же, как и у `GOTO`, однако, в конце подпрограммы необходимо поместить оператор `RETURN`. Он возвращает выполнение оператору, следующему за оператором `GOSUB`, вызвавшим переход в подпрограмму. Подпрограмму можно вызывать из любого места программы, однако есть ограничение на количество вызовов `GOSUB` в программе. Всего, в одной программе, можно использовать 255 вызовов. Также существует ограничение по вложенности подпрограмм (вызов подпрограммы из подпрограммы), максимальная глубина – 4 вызова.

Комментарии

Так же для наглядности исходного кода присутствует возможность вставки в код комментариев. Комментарии могут находиться в любой части программы. Чтобы указать компилятору, что текст является комментарием, необходимо перед текстом поставить апостроф «’». Все что находится от апострофа до конца строки, будет считаться комментарием.

Лабораторные работы

Исполнительные механизмы

Лабораторная работа 1.1 «Управление движением»

Цель работы: обучение управлению сервоприводами цикличного вращения и заданию определенных законов движения роботу в целом.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Сервопривод цикличного вращения – 2 шт.

В работе не используются внешние датчики для получения информации об окружающей среде. К выходам контроллера подключаются только сервоприводы.

Теоретические сведения:

Управление вращением сервопривода осуществляется подачей на него серии импульсов. Скорость и направление вращения зависят от ширины импульсов.

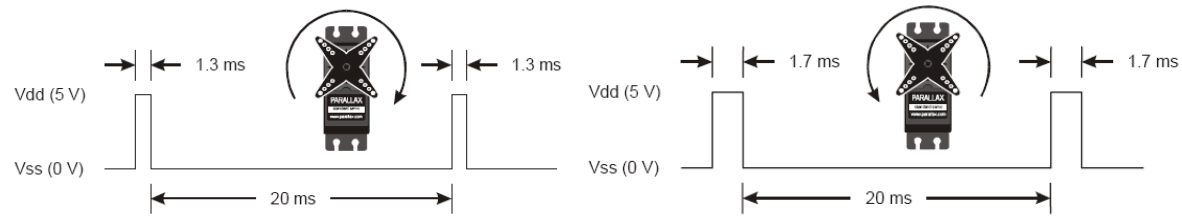
В языке PBasic сгенерировать импульс можно командой `PULSOUT`. Она генерирует 2-х микросекундный импульс, которым и управляется данная модель двигателя. Команда имеет следующий синтаксис:

```
PULSOUT [#], [N]
```

`[#]` – указывает на порт вывода контроллера, к которому подключен двигатель,

`[N]` – ширина импульса.

Именно ширина генерируемого импульсов позволяет управлять скоростью и направлением вращения двигателя. Чем выше ширина импульса, тем больше скорость вращения колеса. Импульсы подаются шириной от 1.3ms до 1.5ms (вращение по часовой стрелке) и от 1.5ms до 1.7ms (вращение против часовой стрелки). При подаче импульсов шириной в 1.5ms вал не вращается. Между импульсами должна быть обязательная пауза, например 20ms. Следует помнить, что команда в программе выполняется гораздо быстрее, чем отработка самого действия, которое задается этой командой, поэтому при подаче `PULSOUT` в цикле необходимо притормаживать ход программы примерно на 20 миллисекунд за один проход. Это можно реализовать с помощью команды `PAUSE`, которая в качестве параметра принимает время (в миллисекундах).



BASIC Stamp Module	1.3 ms	1.5 ms	1.7 ms
BS1	100	150	200
BS2, BS2e, BS2pe	500	750	1000
BS2sx, BS2px, BS2p24/40	1250	1875	2500

Рис. 1. Вращение в разные стороны и таблица ширины импульса в единицах микроконтроллера

Команда `PULSOUT 12, 650` приведет к вращению правого двигателя по часовой стрелке с полной скоростью в течение $650 \cdot 2\mu s$. Задать вращение на более долгий период можно с помощью цикла. Лучше всего использовать цикл с известным числом повторений.

Для того чтобы выяснить, сколько нужно повторений, чтобы двигатель вращался нужное время, необходимо поделить необходимое время вращения на сумму времени вращения двигателя за один проход цикла, паузы в цикле и времени отработки цикла. Время вращения двигателя за один проход цикла находится по формуле $[N] \cdot 0{,}002ms$. Время паузы берется из параметра команды `PAUSE`. Время отработки цикла примерно равно $1{,}5ms$. Следовательно, для конструкции:

```
FOR counter = 1 TO <PULSECOUNT>
 PULSOUT 12, 650
 PAUSE 20
NEXT
```

Для вращения двигателя в течение пяти секунд, значение `PULSECOUNT` будет равняться $\frac{5000}{650 \cdot 0{,}002 + 20 + 1{,}5} \approx 220$.

При необходимости бесконечного вращения двигателем, удобнее использовать конструкцию `DO … LOOP` следующего вида:

```
DO
 PULSOUT 12, 650
 PAUSE 20
LOOP
```

Вращение левым двигателем осуществляется таким же образом, только вместо 12-го порта, следует использовать 13-й. Одновременное вращение двигателями происходит при последовательной подаче команд на 2 выхода. Вследствие того, что двигатели закреплены на разных сторонах корпуса, для задания роботу движения по прямой, двигатели

должны крутиться в разных направлениях, один по часовой стрелке, другой – против.

При необходимости использования нескольких разных циклов для управления двигателем в одной программе, для удобства использования и улучшения читабельности кода, следует использовать процедуры. В качестве параметров, процедуре следует передавать количество импульсов на правый двигатель, количество импульсов на левый двигатель и общее количество повторений цикла. Обращение к процедуре для движения робота вперед в течение пяти секунд, будет выглядеть следующим образом:

```
RPulse = 650
LPulse = 850
MotorCount = 220
GOSUB Motor
```

Процедура Motor:

```
Motor:
FOR counter = 1 TO MotorCount
 PULSOUT 12, RPulse
 PULSOUT 13, LPulse
 PAUSE 20
NEXT
```

Для задания движения по кривой, необходимо задавать несимметричные значения количества импульсов на двигатели. Например, если подавать на один из двигателей количество импульсов, соответствующее вращению с максимальной скоростью, а на второй, количество импульсов, соответствующее остановке двигателя, робот будет крутиться вокруг стоящего колеса. Если подавать на второй двигатель, количество импульсов, соответствующее вращению в ту же сторону, что и первое колесо, но с меньшей скоростью, то робот будет двигаться по дуге.

Порядок выполнения работы:

1. Подать на оба двигателя 750 импульсов в течение нескольких секунд и убедиться, что двигатели не вращаются.
2. Изменяя количество проходов цикла, подобрать нужные значения для отработки действий, указанных в таблице 1.1. Заполнить таблицу.
3. Написать программы для описания роботом следующих траекторий:
   - Квадрат, с выполнением поворотов различными способами.
   - Траекторию, изображенную на рисунке 1.1.

Таблица 1.1

Действие	Импульсы на двигатели		**Кол-во проходов**
	Левый	**Правый**	
Поворот колес на один оборот по часовой стрелке с полной скоростью.			
Движение вперед на расстояние 10 см с полной скоростью.			

Поворот робота на 90º направо, вращением левого колеса с полной скоростью.			
Поворот робота на 45º налево, вращением правого колеса с полной скоростью.			
Поворот робота на 90º направо, вращением колес в разных направлениях с полной скоростью.			
Разворот робота на 180º, вращением колес в разных направлениях с полной скоростью.			
Проезд по окружности с возвращением в точку отправления, вращением обоих колес в одну сторону, но с разной скоростью.			

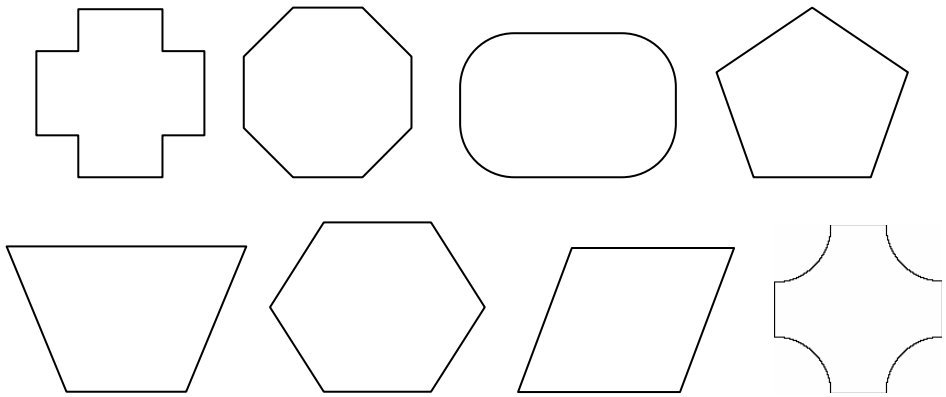


Рис. 1.1. Варианты траекторий для заданий

Лабораторная работа 1.2
«Управление позиционным сервоприводом»

Цель работы: обучение управлению позиционными сервоприводами.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2
- Позиционный сервопривод – 1 шт.

В работе не используются внешние датчики для получения информации об окружающей среде. К выходам контроллера подключается только шаговый двигатель.

Теоретические сведения:

Управление вращением двигателя сервопривода осуществляется подачей на него серии импульсов. Двигатель, использующийся в данной лабораторной работе, позволяет выставить любой угол от 0° до 180°. Каждое значение градуса задается шириной импульса.

В языке PBasic сгенерировать импульс можно командой `PULSOUT`. Она генерирует 2-х микросекундный импульс, которым управляется данная модель двигателя. Команда имеет следующий синтаксис:

```
PULSOUT [#], [N]
```

`[#]` – указывает на порт вывода контроллера, к которому подключен двигатель,

`[N]` – ширина импульса.

Именно ширина генерируемого импульса позволяет выставлять нужный нам угол. Для используемого в работах двигателя, значения этого параметра должно находиться между 250 (0°) и 1150 (180°.). Значение 700 соответствует 90°.

Команда `PULSOUT 14, 750` приведет к вращению двигателя к положению 90° в течении $750 \cdot 2\mu s$. Задать вращение на более долгий период можно с помощью цикла. Лучше всего использовать цикл с известным числом повторений. Следует помнить, что команда в программе выполняется гораздо быстрее, чем отработка самого действия, которое задается этой командой, поэтому при подаче `PULSOUT` в цикле необходимо притормаживать ход программы примерно на 20 миллисекунд за один проход. Это можно реализовать с помощью команды `PAUSE`, которая в качестве параметра принимает время (в миллисекундах).

Пример выставления определенного угла и удержания его в течение пяти секунд (расчет времени см. лаб. раб. 1.1):

```
FOR counter = 1 TO 220
  PULSOUT 14, 750
  PAUSE 20
NEXT
```

При необходимости использования нескольких разных циклов для управления двигателем в одной программе, для удобства использования и улучшения читабельности кода, следует использовать процедуры. В качестве параметров, процедуре следует передавать количество импульсов на двигатель и общее количество повторений цикла.

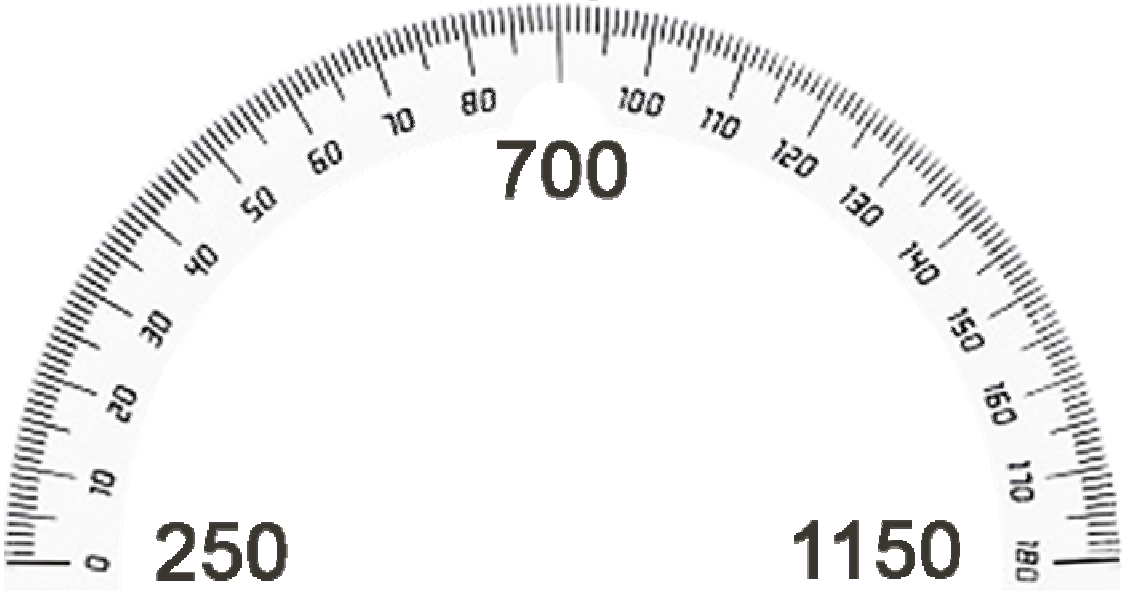


Рис. 1.2 Шкала соответствия градусов и импульсов

Порядок выполнения работы:

1. Подать на сервопривод 750 импульсов в течение нескольких секунд и убедиться, что двигатель устанавливается в положение, равное 90°.
2. Написать программу для прохождения сервоприводом углов от 0° до 180° с шагом в 10°. Заполнить таблицу 1.2

Таблица 1.2

Значение угла	Ширина импульсов	Кол-во проходов
0°	250	
10°		
20°		
30°		
40°		
50°		
60°		
70°		
80°		
90°	700	
100°		
110°		
120°		
130°		
140°		
150°		
160°		
170°		
180°	1150	
90°	700	

Датчики и сенсоры

«Использование инфракрасного излучения для определения объекта»

Общие сведения

Для определения объекта можно использовать, специальные средства, при помощи которых робот видит объект непосредственно при контакте. Наиболее простая система обнаружения объекта это направить на объект инфракрасное излучение при помощи инфракрасного излучателя и поймать отраженное от объекта излучение при помощи приемника инфракрасного излучения. Благодаря широкому распространению они являются недорогими и легкодоступными средствами.

Инфракрасное излучение — электромагнитное излучение, занимающее спектральную область между красным концом видимого света (с длиной волны $\lambda = 0{,}74$ мкм) и микроволновым излучением ($\lambda \sim 1$—2 мм).

Сейчас весь диапазон инфракрасного излучения подразделяют на три составляющие:

коротковолновая область: λ=0,75 - 1,5 мкм;

средневолновая область: λ=1,5 – 5,6 мкм;

длинноволновая область: λ=5,6 - 100 мкм;

Последнее время длинноволновую окраину этого диапазона выделяют в отдельный, независимый диапазон электромагнитных волн — терагерцовое излучение (субмиллиметровое излучение).

Инфракрасное излучение также называют «тепловым» излучением, так как все тела, твёрдые и жидкие, нагретые до определённой температуры, излучают энергию в инфракрасном спектре. При этом длины волн, излучаемые телом, зависят от температуры нагревания: чем выше температура, тем короче длина волны и выше интенсивность излучения. Спектр излучения абсолютно чёрного тела при относительно невысоких (до нескольких тысяч градусов Кельвин) температурах лежит в основном именно в этом диапазоне.

Использование

ИК (инфракрасные) диоды и фотодиоды повсеместно применяются в пультах дистанционного управления, системах автоматики, охранных системах и т.д. Практически полное вытеснение красных излучателей из этой области объясняется тем, что они не отвлекают и не привлекают внимание человека в силу своей невидимости. Инфракрасные излучатели применяют в промышленности для сушки лакокрасочных поверхностей. Инфракрасный метод сушки имеет существенные преимущества перед традиционным, конвекционным методом. В первую очередь это, безусловно, экономический эффект. Скорость и затрачиваемая энергия при инфракрасной сушке в разы меньше тех же показателей затрачиваемых при традиционных методах. Положительным побочным эффектом так же является стерилизация продуктов питания, увеличение стойкости к коррозии покрываемых красками поверхностей. Недостатком же является существенно большая неравномерность нагрева, что в ряде технологических процессов совершенно неприемлемо. Особенностью применения ИК-излучения в пищевой промышленности является возможность проникновения электромагнитной волны в такие капиллярно-пористые продукты, как зерно, крупа, мука и т.д. на глубину до 7мм. Эта величина зависит от характера поверхности, структуры, свойств материала и частотной характеристики излучения. Электромагнитная волна определённого частотного диапазона оказывает не только термическое, но

и биологическое воздействие на продукт, способствует ускорению биохимических превращений в биологических полимерах (крахмал, белок, липиды). Конвейерные сушильные транспортёры с успехом могут использоваться при закладке зерна в зернохранилища и в мукомольной промышленности.

Цвет и приблизительная длина волны:

Цвет	**Длина волны** [нм]	**ИК**	**Длина волны** [мкм]
Фиолетовый	380-435	*Ближ. инфр. изл.*	0,75 - 1,5
Синий	435-500	*Инфракрасное*	1,5 - 5,6
Зеленый	500-555	*Дал. инфр. изл.*	5,6 - 100
Желтый	555-600		
Оранжевый	600-650		
Красный	650-780		

Используемые в нашей лабораторной работе инфракрасный излучатель и приемник работают с излучением длинной 980нм.

Инфракрасная система обнаружения объекта похожа в некотором отношении на фары автомобиля. (Рис. 2.1.) Когда свет фар автомобиля отражается от препятствия, наши глаза это улавливают и передают информацию в мозг, а мозг соответственно дает команды органам управления, вести автомобиль правильно. Boe–Bot использует инфракрасные светодиоды в качестве фар, а глаза – это приемники инфракрасного излучения. Инфракрасные приемники излучения посылают сигналы в микроконтроллер, действительно ли они обнаружили инфракрасный отраженный сигнал от объекта или нет. А микроконтроллер уже в зависимости от заданной программы посылает соответствующие сигналы на другие устройства, например двигатели.

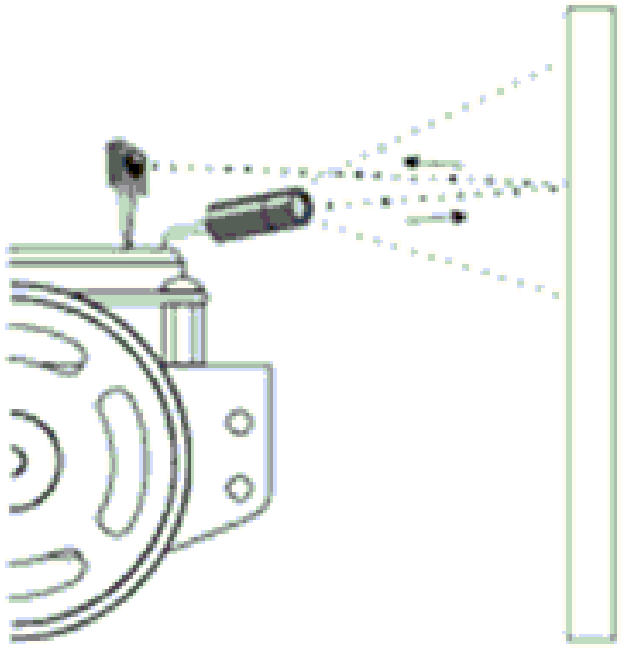


Рис. 2.1. Обнаружение объекта

Лабораторная работа 2.1
«Регулирование дистанции обнаружения объектов изменением частоты излучения»

Цель работы: определить зависимость интенсивности излучения инфракрасного диода от частоты, на которой он излучает.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2
- Приемник инфракрасного излучения;

Рис. 2.2 Приемник инфракрасного излучения

- Инфракрасный светодиод;

Рис. 2.3 Инфракрасный светодиод

- Корпус для инфракрасного светодиода;

Рис. 2.4 Корпус

- Резистор на 220 Ом (цветовая маркировка: красный-красный-коричневый);
- Резистор на 1 кОм (цветовая маркировка: коричневый-черный-красный);

Собранный излучатель ИК выглядит, как показано на рисунке 2.5.

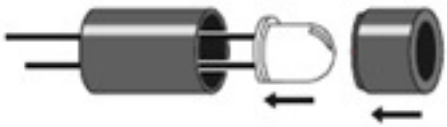


Рис. 2.5 Светодиод в корпусе

Теоретические сведения:

Пара датчиков должна быть установлена в углу макетной платы. На рисунке 2.6 показана электрическая схема подключения датчиков.

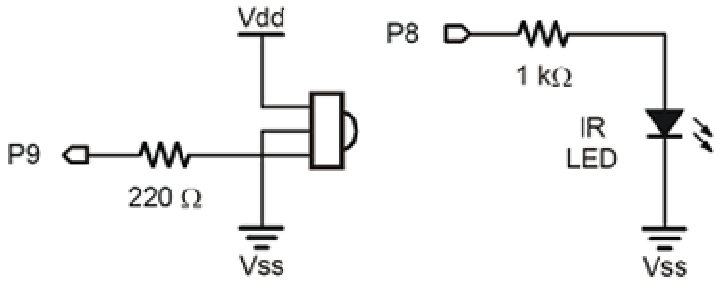


Рис. 2.6 Схема подключения

Особое внимание нужно уделить правильности установки инфракрасного диода. Чтобы обеспечить работоспособность не путать местами аноды и катоды. Более длинная ножка это анод, он должен подключатся через сопротивление. Более короткая ножка это катод, он подключается к “-“ (VSS).

Тестирование датчиков

Работа с инфракрасным диодом и приемником. Для того чтобы приемник не реагировал на сторонние инфракрасные излучения необходимо посылать такой сигнал, который бы отличался от них. Будем использовать синусоидальный сигнал с частотой 38,5 kHz . Для генерации синусоиды используется команда

```
FREQOUT [#], [T], [F]
```

`[#]` – номер порта, к которому подключен ИК диод.

`[T]` – продолжительность синусоиды в миллисекундах.

`[F]` – частота.

Если ИК-приемник примет отраженный сигнал, он изменит напряжение на своем контакте, отвечающем за обнаружение сигнала. Получить значение напряжения на этом контакте, можно с помощью порта контроллера, к которому подключен этот контакт. Зная, какое напряжение соответствует наличию сигнала, а какое – отсутствию, можно сделать вывод о наличии либо отсутствии препятствия.

Процесс считывания информации заключается в чтении значения порта ввода, к которому подключен приемник. Делается это с помощью глобальной константы, отвечающей за состояние портов.

`IN#` - где `#`, порт ввода.

Считывать состояние порта необходимо сразу после отсылки сигнала диодом и записывать его в переменную. Для левого ИК диода и ИК-приемника это выглядит так:

```
FREQOUT 8, 1, 38500
IR = IN9
```

При отсутствии сигнала, значение `IR` будет равно 1, при наличии сигнала - 0.

Пример программы для тестирования датчиков

```
MAIN:
  FREQOUT 7, 1, 38500
  IR = in8
  IF IR = 1 then unbroken
    DEBUG HOME, "Beam is broken; object detected."
    GOTO MAIN
  unbroken:
    DEBUG HOME, "Beam is unbroken; object not detected."
    GOTO MAIN
```

Если перед датчиком поставить препятствие он сообщит о его обнаружении.

Используя это принцип, в данном задании нужно будет изменять значения частоты на ИК светодиоде командой `FREQOUT`, перебирая их от 38.5 кГц до 45 кГц.

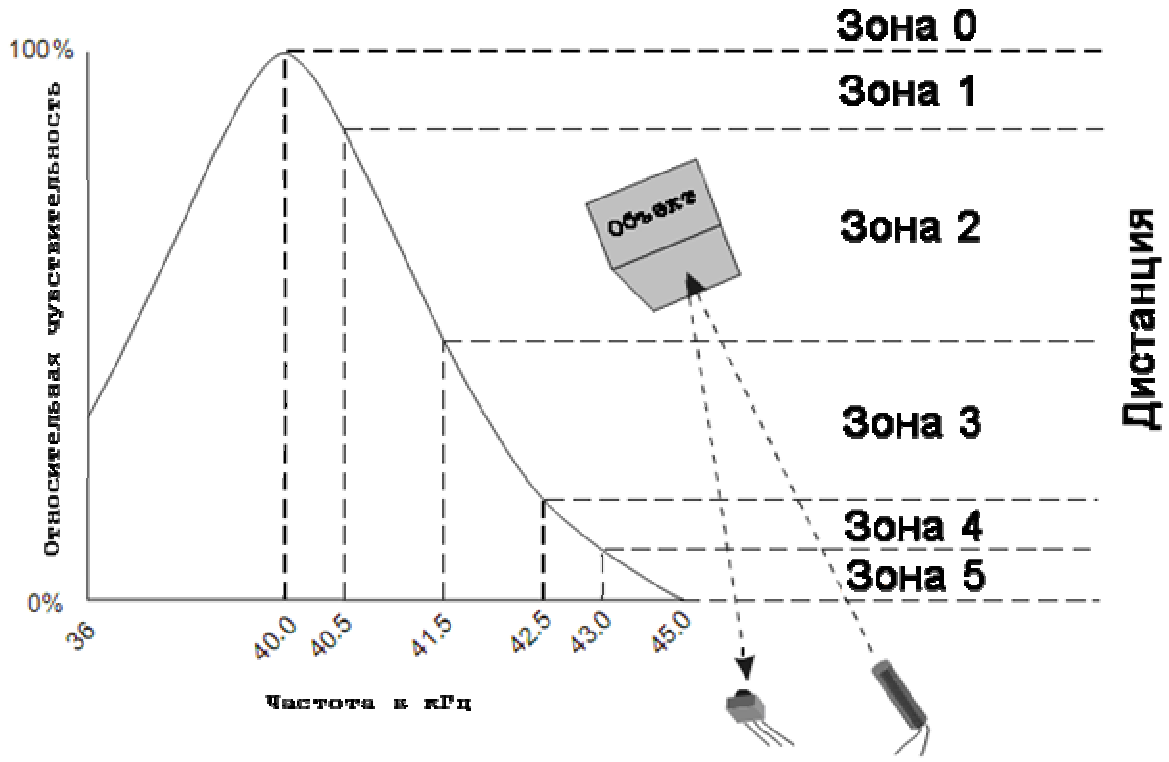


Рис. 2.7 Зоны чувствительности

Порядок выполнения работы:

1. Написать программу для работы с ИК датчиками и загрузить ее в микроконтроллер.
2. Изменяя значения частоты, измерить расстояния зон. Заполнить таблицу.

Таблица 2.1

Значение частоты [Гц]	**Максимальное расстояние** [см]
37500	
40000	
40500	
41500	
42500	
43000	
45000	

Лабораторная работа 2.2
«Регулирование дистанции обнаружения объектов изменением тока в ИК диоде»

Цель работы: определить зоны чувствительности ИК датчика, регулируя ток, протекающий в ИК диоде.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2
- Приемник инфракрасного излучения
- Инфракрасный светодиод
- Резистор 220 Ом (красный-красный-коричневый)
- Резистор 470 ОМ (желтый-фиолетовый-коричневый)
- Резистор 1000 Ом (коричневый-чёрный-красный-золотой)
- Резистор 2000 Ом (красный-черный-красный)
- Резистор 4700 Ом (желтый-фиолетовый-красный)

Теоретические сведения:

Принцип этого задания основан на изменении интенсивности освещения.

Используя это принцип, будем менять значения сопротивления на ИК светодиоде, уменьшая и увеличивая его. И сделаем соответствующие выводы о зоне чувствительности в зависимости от силы тока в ИК диоде.

Внимание! Чтобы избежать повреждения оборудование, перед каждой заменой резистора необходимо выключать питание платы. Выключатель переводить в положение «0» - питание отключено.

Порядок выполнения работы:

1. Загрузить в микроконтроллер программу из предыдущей лабораторной работы.
2. Поочередно меняя сопротивления в цепи светодиода, измерять расстояния определения объекта. Заполнить таблицу.

(Значение частоты – 37500 Гц)

Таблица 2.2

Сопротивление [Ом]	**Максимальная определяемая дистанция** [см]
4700	
2000	
1000	
470	
220	

«Использование ультразвукового излучения для определения объекта»

Общие сведения

Ультразвук распространяется в средах в виде чередующихся зон сжатия и расширения молекул вещества, которые совершают колебательные движения. Звуковые волны, в том числе и ультразвуковые, характеризуются периодом колебания — временем, за которое молекула (частица) совершает одно полное колебание; частотой — числом колебаний в единицу времени; длиной — расстоянием между точками одной фазы и скоростью распространения, которая зависит главным образом от упругости и плотности среды. Длина волны обратно пропорциональна её частоте. Чем меньше длина волн, тем выше разрешающая способность ультразвукового аппарата.

Человеческое ухо воспринимает распространяющиеся в среде упругие волны частотой приблизительно до 16-20 кГц; колебания с более высокой частотой представляют собой ультразвук (за пределом слышимости). Обычно ультразвуковым диапазоном считают полосу частот от 20 000 до нескольких миллиардов Гц.

Лабораторная работа 3.1
«Определение расстояния до объекта ультразвуковым датчиком»

Цель работы: измерение расстояния до объекта ультразвуковым датчиком.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Ультразвуковой датчик PING.

Теоретические сведения:

Ультразвуковой модуль PING используется для измерения расстояния для объекта от 3 см до 3 м.

Рис. 3.1 Внешний вид УЗ датчика PING

На рисунке 3.2 наглядно изображено, как датчик посылает ультразвуковой импульс и измеряет время возвращения отраженного сигнала, регистрируемого приемником. Частота ультразвукового сигнала датчика PING составляет 40 кГц.

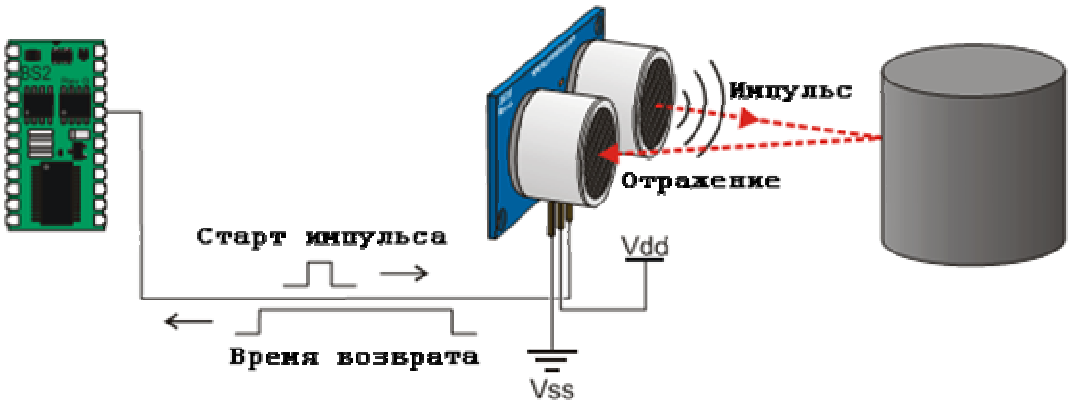


Рис. 3.2 Принцип действия

Команды для работы с датчиком: PULSOUT (инициирует излучение) и PULSIN (принимает излучение). Микроконтроллер в свою очередь может измерить время между излучением сигнала и его возвратом. Используя эти данные и скорость звука в воздухе, можно вычислить расстояние до объекта.

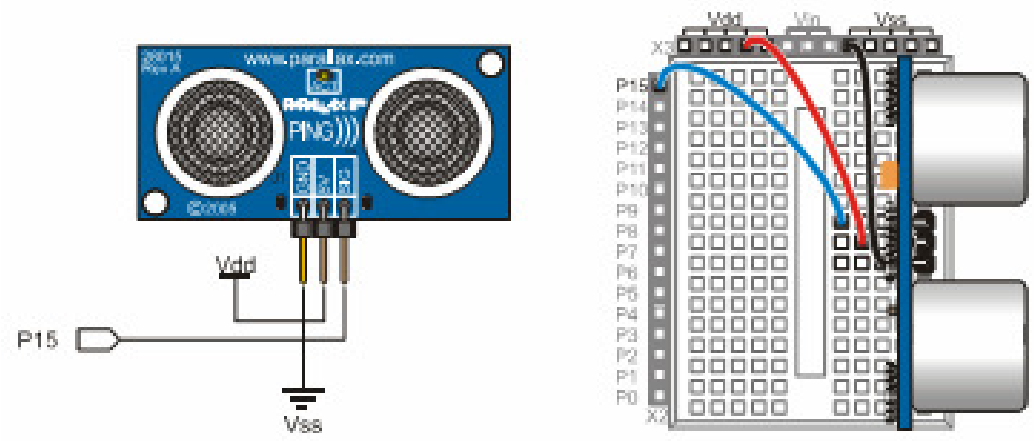


Рис. 3.3 Подключение к плате

В языке PBasic сгенерировать последовательность импульсов можно командой `PULSOUT`. Она генерирует 2-х микросекундные импульсы, которыми управляется данная модель датчика. Команда имеет следующий синтаксис:

`PULSOUT [#], [N]`

`[#]` – указывает на порт вывода контроллера, к которому подключен датчик,

`[N]` – количество импульсов,

`PULSIN [#], [H], [T]`

`[#]` – указывает на порт вывода контроллера, к которому подключен датчик,

`[N]` – может быть либо 0, либо 1, указывает на то, 0 или 1 считать сигналом,

`[T]` – переменная, означающая измеренную длительность импульса.

Расчет расстояния.

Формула для расчета дистанции:

$S = C_{возд} \cdot t$, где $C_{возд}$ - скорость распространения звуковой волны в воздухе, t – время. Так как измеренное время состоит из времени следования волны до объекта и времени возврата, то расстояние до объекта определяется по формуле:

$$S_{об} = \frac{S}{2} = \frac{C_{возд} \cdot t}{2}$$

Скорость звука измеряется в м\с. Для получения значений в *см* умножим расстояние до объекта на 100:

$$S_{об} = \frac{100 \cdot C_{возд} \cdot t}{2}$$

Длительность импульса, который принимает команда PULSIN, равняется 2/1,000,000 в секунду (2 микросекунды). Вместо t в секундах, подставим в формулу время микроконтроллера:

$$S_{об} = \frac{100 \cdot C_{возд} \cdot t_{PULSIN}}{2} \cdot \frac{2}{1000000}$$

В итоге формула примет вид:

$$S_{об} = \frac{C_{возд} \cdot t_{PULSIN}}{10000}$$

Скорость звука в воздухе при температуре 72 °F (22.2 °C) равна 344.8 м/с:

$$S_{об} = \frac{344.8 \cdot t_{PULSIN}}{10000} = 0.03448 \cdot t_{PULSIN}$$

В переменной `PULSIN` храниться дробное значение, следовательно нужно использовать оператор «**» для перемножения дробных значений

меньше единицы. Найдем эквивалент константы, подходящий для вычислений в Stamp.

$$\text{CmConstant} = 0.03448 \cdot 65536 = 2259{,}68 \approx 2260$$

Порядок выполнения работы:

1. Написать программу для измерения расстояния до объекта, используя конструкцию:

```
PULSOUT 15, 5
PULSIN  15, 1, time
```

В переменной time будет храниться время измерения. Для пересчета в сантиметры используйте конструкцию:

```
distance = time ** 2260
```

2. Значения time и distance выводите в окно отладки.

```
DEBUG CLS
DEBUG HOME, "time = ", DEC5 time
DEBUG HOME, "distance = ", distance
```

3. С помощью получившегося ультразвукового дальномера заполните таблицу.

Таблица 3.1

Точное расстояние [см]	**Измеренное расстояние** [см]	**Время**
10		
20		
30		
40		
50		
60		
70		

Лабораторная работа 3.2
«Сканирование окружающего пространства»

Цель работы: научиться использовать ультразвуковой датчик и поворотное устройство как единый мехатронный узел.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Позиционный сервопривод;
- Ультразвуковой датчик PING.

Порядок выполнения работы:

1. Написать программу, которая будет поворачивать датчик от 0° до 180°, с шагом 10°;
2. Измерить расстояния в этих углах. Можно использовать программы, написанные в предыдущих работах. Значения

выводить в окно отладки. Используйте команду PAUSE для того, чтобы успевать записывать значения.

3. Заполните таблицу.

Таблица 3.2

Значение угла	Расстояние
0°	
10°	
20°	
30°	
40°	
50°	
60°	
70°	
80°	
90°	
100°	
110°	
120°	
130°	
140°	
150°	
160°	
170°	
180°	

Лабораторная работа 3.3
«Определение зон нечувствительности датчика»

Цель работы: исследование ситуаций, в которых ультразвуковой датчик может быть неэффективен, либо выдавать ложные значения.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Позиционный сервопривод;
- Ультразвуковой датчик PING.

На рисунке 3.4 изображены ситуации, в которых датчик может быть неэффективен.

Порядок выполнения работы:

1. Найти максимальное расстояние, которое может измерить датчик.
2. Определить размер объекта, который датчик не может «увидеть», на расстоянии 50 см, 1 м, 2 м.
3. Определить углы, на которых датчик неэффективен. Написать соответствующую программу.

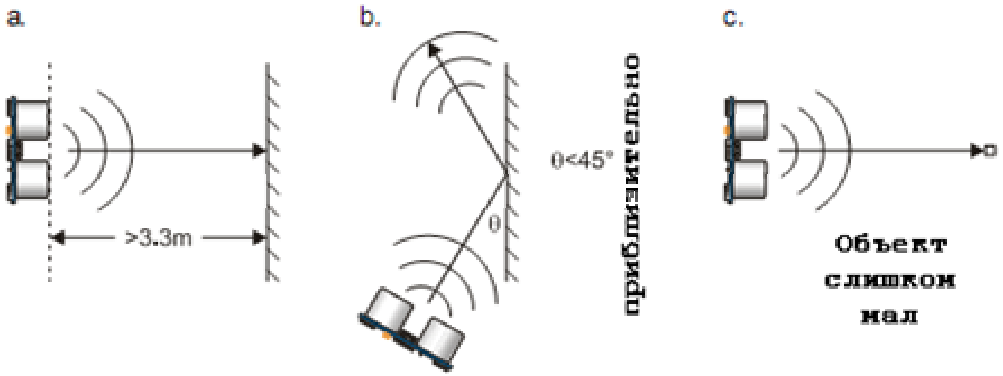


Рис. 3.4 Зоны нечувствительности датчика

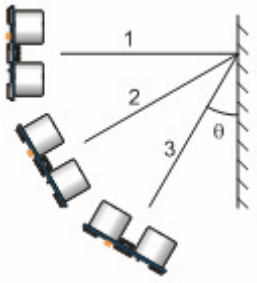


Рис. 3.5 Угол отражения

Таблица 3.3

Значение угла	Наличие отражения сигнала
0°	
10°	
20°	
30°	
40°	
50°	
60°	
70°	
80°	
90°	

«Использование MEMS датчиков для навигации»

Общие сведения

Микроэлектромеханические системы (МЭМС) — технологии и устройства, объединяющие в себе микроэлектронные и микромеханические компоненты. МЭМС устройства обычно изготавливают на кремниевой подложке с помощью технологии микрообработки, аналогично технологии изготовления однокристальных

интегральных микросхем. Типичные размеры микромеханических элементов лежат в диапазоне от 1 микрометра до 100 микрометров, тогда как размеры кристалла МЭМС микросхемы имеют размеры от 20 микрометров до одного миллиметра.

МЭМС технологии применяются для создания разнообразных миниатюрных датчиков, таких как акселерометры, датчики угловых скоростей, магнитометрические датчики, барометрические датчики, анализаторы среды (например для оперативного анализа крови).

МЭМС технология может быть реализована с использованием целого ряда различных материалов и технологий производства, выбор которых будет зависеть от создаваемого устройства и рыночного сектора, в котором он должен работать.

Кремний является материалом, используемым для создания наиболее интегрированных цепей, используемых в потребительской электроники в современном мире. Распространенность, доступность дешевых высококачественных материалов и способность к проявлению электронных функций делает кремний привлекательным для применения его в широком кругу МЭМС. Кремний также имеет значительные преимущества, благодаря свойства самого материала. Из-за единой структуры кристалла, кремний является почти идеальным материалом, это означает, что, когда он согнут, фактически нет гистерезиса и, следовательно, почти нет утечки энергии. А также кремний очень надежен при сверхчастых движениях, так как он обладает очень малой усталостью и может работать в диапазоне от миллиардов до триллионов циклов без разрушения.

Лабораторная работа 4
«Двухосевой MEMS компас»

Цель работы: применение двухосевого MEMS компаса в подвижных системах.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Сервопривод цикличного вращения – 2 шт;
- Компас Hitachi HM55B.

Теоретические сведения:

Как сориентировать на месте небольшие мобильные устройства, такие как подвижные роботы, автоматические транспортные средства, перемещающиеся в небольшом замкнутом пространстве, где расстояния их перемещения равны или меньше точности измерения спутниковых систем (десятки и единицы метров)? Использовать компас. Но магнитный компас представляет собой замкнутую механическую систему, не имеющую

выходных электрических сигналов. Одним из решений является применение двухосевого MEMS датчика Hitachi HM55B.

Данный компас позволяет измерить силу магнитного поля Земли по осям x и y. Значение x – это компонент магнитного поля, воздействующий на ось x датчика. Значение y – это отрицательная составляющая магнитного поля Земли, воздействующая на ось y датчика. На рисунке изображены геометрические соотношения, позволяющие нам составить математические формулы, необходимые для вычисления значений.

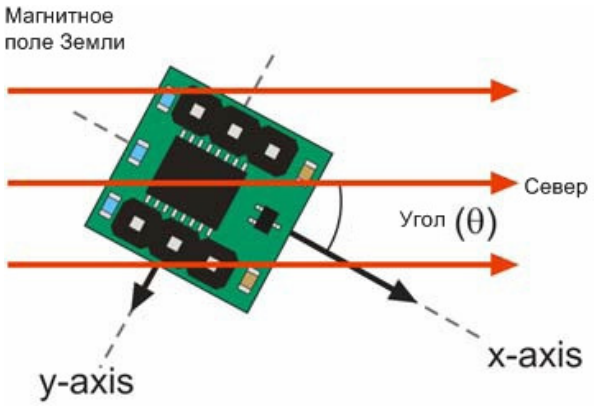
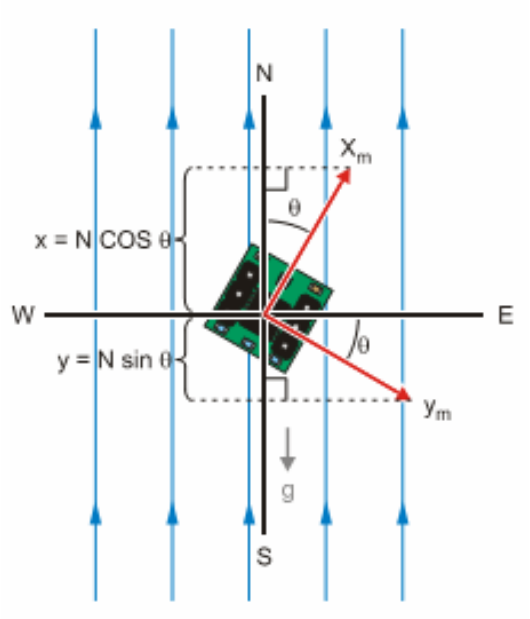


Рис. 4.1 Принцип работы HM55B

Характеристики модуля:

Чувствительность к изменению напряженности магнитного поля

- Чувствительность к изменению напряженности магнитного поля на уровне микротесла (uT): 1.0uT/lsb…1.6uT/lsb
- Линейный диапазон измерений: ±180uT
- Определение направления магнитных линий с помощью чувствительных элементов по двум осям
- Разрешение компаса после калибровки: 6bit (64-направления, то есть одно значение шкалы - 4 угловых градуса)
- Время выдачи готовых данных: 30ms…40ms
- Простой 3-проводный синхронный последовательный интерфейс
- Встроенный согласующий резистор для устранения риска конфликтов на цифровой шине
- Напряжение питания: 5V
- Встроенный стабилизатор напряжения : 3V
- Ток потребления: 5mA…7mA
- Пиковый ток потребления: 30mA…45mA
- Ток потребления в режиме ожидания: 2mA…3mA
- Предельная напряженность поля: 300uT



$$\tan\theta = \frac{-N\cdot\sin\theta}{N\cdot\cos\theta} = \frac{-y}{x}$$

$$\tan^{-1}(\tan\theta) = \tan^{-1}\left(\frac{-y}{x}\right)$$

$$\theta = \tan^{-1}\left(\frac{-y}{x}\right)$$

Рис. 4.2 Геометрические соотношения

Модуль компаса представляет из себя печатную плату с двухрядными штыревыми разъёмами (DIP-6) и работает в диапазоне температур 0°C…+70°C.

Рис. 4.3 Внешний вид датчика

Подключение к микроконтроллеру.

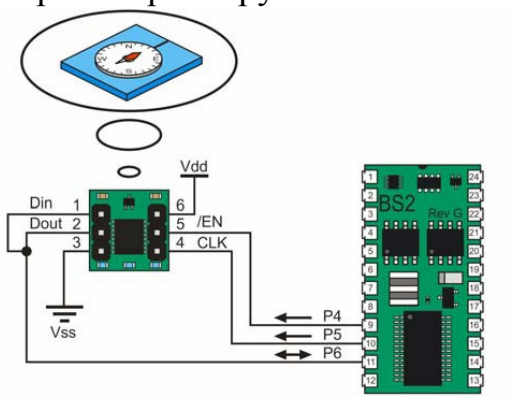


Рис. 4.4 Подключение к микроконтроллеру

Во время проведения измерений вы заметите, что значение x и y при неизменном положении датчика будут чередоваться между 2-4 величинами. Это результат воздействия различных помех, которые в сумме называются шумом измерения. Некоторые элементы, создающие помехи, это питающие детали, линии электропередач, цифровая активность микроконтроллера и даже цифровая активность в самом HM55B.

Один из эффективных способов устранения шумов, это усреднение значений x и y.

Порядок выполнения работы:

1. Загрузить в микроконтроллер программу и запустить «HM55B Demo.exe». Убедиться в работоспособности датчика.
2. Написать программу движения робота по траектории из лабораторной работы №1.

В контрольных точках, указанных преподавателем вывести данные компаса.

«Датчики количества оборотов, как элементы навигационной системы робота»

Общие сведения

В настоящее время большинство роботов оснащены одним или несколькими двигателями вращательного движения. Для того чтобы оперативно контролировать угол поворота вала двигателя часто на него устанавливается дисковый датчик. Этот датчик состоит из диска с отверстиями (устанавливается на вал) и декодера, устройства распознания отверстий (устанавливается на двигатель).

Зная угол поворота вала двигателя, становится возможным рассчитать пройденный путь, а так же скорость вращения вала.

В данных лабораторных работах используется набор инфракрасных энкодеров «Digital encoder kit» для робота Boe-Bot.

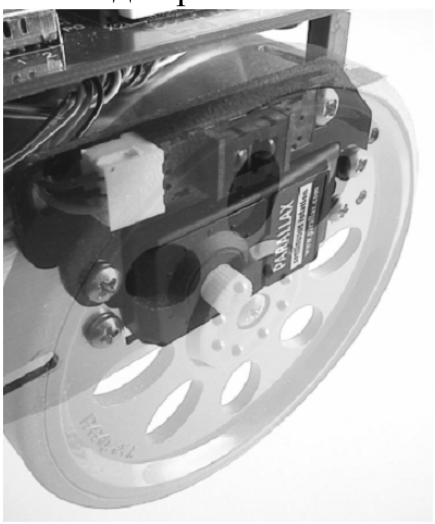


Рис. 5.1 Установленный энкодер

Состав комплекта «Digital encoder kit».

1. Рефлексивный сенсор - 2 шт.

Принцип работы основан на посылке ИК сигнала и его отражении в приемник.

Рис. 5.2 Внешний вид блока ИК излучателя и приемника

2. Соединительный кабель - 2 шт.

Рис. 5.3 Внешний вид соединительных кабелей

3 Монтажная панель - 2 шт.

Рис. 5.4 Внешний вид монтажной панели

4. Винт, гайка резистор (10 кОм) – по 2 шт.

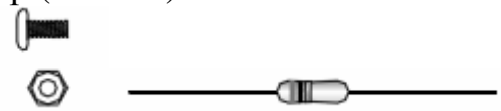


Рис. 5.5 Внешний вид крепежа и резистора

Подключение к плате:

`P11` и `P10` – входы на плате, в которые подаются сигналы с датчиков.

Если сенсор «видит» предмет перед собой то на вход подается низкое напряжение, если нет то высокое. Соответственно 0 или 1.

Лабораторная работа 5.1

«Измерение длины окружности колеса»

Цель работы: измерить длину окружности колеса и длину сектора.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Сервопривод цикличного вращения – 2 шт.
- Энкодер – 2 шт.

Теоретические сведения:

Один оборот колеса равен 16ти импульсам энкодера.

Регистрировать импульсы энкодера лучше всего с помощью счетчика.

Считывание нового значения осуществляется конструкцией:

```
New = INS.LOWBIT(PIN #)
IF (New ^ Prev) THEN
```

```
Prev = New
Counter = Counter + 1
```

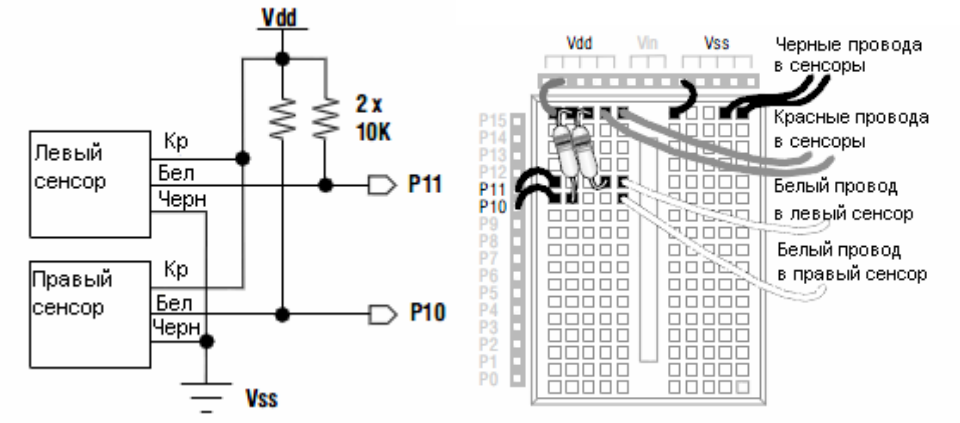


Рис. 5.6 Визуальное изображение подключения к плате

Порядок выполнения работы:

1. Открыть программу encoder.bs2 и загрузить ее в микроконтроллер;
2. Положить вдоль пути робота линейку и замерить пройденное расстояние за 1 оборот;
3. Рассчитать значение для 1 сектора;
4. Сделать вывод о совпадении с эталоном (0,5’’=1,27 см).
5.

Лабораторная работа 5.2
«Калибровка с помощью энкодера»

Цель работы: установить соответствие длине импульса двигателя и числу оборотов колеса в минуту.

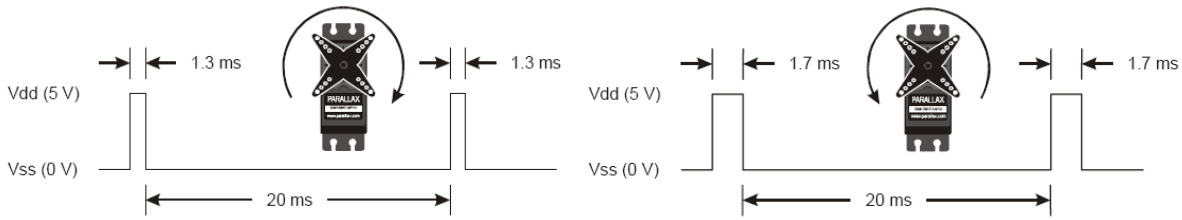


Рис. 5.8 Вращение в разные стороны

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Сервопривод цикличного вращения – 2 шт.
- Энкодер – 2 шт.
- Подставка для робота.

Теоретические сведения:

Так как наш робот использует для передвижения сервоприводы, то для задания оборотов нужно подавать на них импульсы различной длины. Чем больше длина импульса, тем больше скорость вращения колеса. Импульсы

подаются с длиной волны от 1.3ms до 1.5ms(вращение по часовой стрелке) и от 1.5ms до 1.7ms(вращение против часовой стрелки). При подаче импульсов величиной в 1.5ms вал не вращается. Между импульсами должна быть обязательная пауза, например 20ms.

Величина импульса задается с помощью команды PULSOUT. Для получения длины импульса в ms необходимо умножить значение PULSOUT'а на 0.002ms. Каждому значению импульса соответствует свое значение скорости (обороты колеса в минуту).

Рис. 5.9 Зависимость скорости вращения колеса от заданного импульса

Порядок выполнения работы:

1. Установить робота на подставке так, чтобы колеса могли совершать вращение свободно, не касаясь поверхности.
2. Открыть программу encoder2.bs2 и загрузить ее в микроконтроллер. Программа будет предлагать ввести значения импульса, после чего колесо совершит некоторое число оборотов в течение 6ти секунд, затем будут получены данные энкодера.
3. Заполнить таблицу. Значения импульсов получить путем пересчета Времени одного прохода по формуле, зная, что *750*0,002мс=1,5мс.* Указывать число оборотов в минуту. Для этого полученное число оборотов умножить на 10.
4. Построить график отношения величины импульса двигателя к оборотам колеса в минуту.
5. Сравнить теоретический и практический графики.

Таблица 5.2

Время одного прохода	**Значение энкодера**	**Число оборотов**	**Время одного прохода**	**Значение энкодера**	**Число оборотов**
1.300			1.500		
1.310			1.510		

1.320			1.520		
1.330			1.530		
1.340			1.540		
1.350			1.550		
1.360			1.560		
1.370			1.570		
1.380			1.580		
1.390			1.590		
1.400			1.600		
1.410			1.610		
1.420			1.620		
1.430			1.630		
1.440			1.640		
1.450			1.650		
1.460			1.660		
1.470			1.670		
1.480			1.680		
1.490			1.690		
			1.700		

«Датчики изображения и системы технического зрения»

Общие сведения

В последнее время все более актуальными становятся так называемые адаптивные системы, способные быстро и легко перенастраиваться для решения различного рода задач, позволяющие добиться большей эффективности и более широких возможностей. Для наделения объекта адаптивными возможностями требуются современные технологические решения. Одним из таких решений являются системы технического зрения. Техническое зрение подразумевает распознавание реальных объектов на изображении и определение свойств этих объектов, что позволяет решать одновременно несколько задач контроля с высокой скоростью, точностью и надежностью. Зрение в этом смысле делает такие задачи, как определение объектов, ориентация в пространстве и т.д., гораздо более эффективными, а также позволяет снизить затраты на большое количество не интеллектуальных датчиков.

Комплекс устройств, которые входят в состав системы, включает в себя определенный набор технических средств. Основными являются камера, осуществляющая захват изображения, и блок обработки изображения или контроллер. Дополнительными устройствами, но не менее важными, являются оптика, определяющая границы обзора камеры, подсветка, освещающая объект наблюдения, и, если требуется, дисплей для отображения информации в реальном времени и для настройки

системы. Эти компоненты могут совмещаться. Например, камера может иметь встроенный объектив с автофокусировкой и интеллектуальную подсветку. А контроллер или блок обработки может иметь встроенный дисплей. Таким образом, обеспечивается компактность системы.

В зависимости от функционала и сложности устройства делятся на датчики изображения и системы технического зрения. Первые отличаются простотой в настройке и использовании, а также небольшим набором функций. Вторые предоставляют широкий выбор инструментов для решения большого круга задач и содержат сложное программное обеспечение.

Лабораторная работа 6
«Получение и обработка информации видеокамеры»

Цель работы: обучение работе с видеокамерой «CMUCam1 AppMod Vision System» и использованию ее в качестве системы слежения за внешним объектом.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Сервопривод цикличного вращения – 2 шт;
- Видеокамера CMUCam1 AppMod Vision System.

Теоретические сведения:

Используемая в данной лабораторной работе видеокамера имеет разрешение 80x143 пикселей, частота обновления экрана – 17 кадров в сек.

Управление камерой осуществляется через последовательный интерфейс. Для работы с серийным интерфейсом в языке PBasic существуют специальные команды `SERIN`, `SHIFTIN` (для приема данных) и `SEROUT`, `SHIFTOUT` (для отправки данных). Отличие этих команд между собой в том, что первые (`SERIN` и `SEROUT`) передают данные в асинхронном режиме, а вторые (`SHIFTIN` и `SHIFTOUT`) в синхронном.

Управление работой светодиодов происходит с помощью отправки на специальный порт камеры специального числового кода, который указывает на то, какие диоды должны быть включены. Каждому диоду соответствует свое число (табл. 6.4.), если необходимо чтобы работало несколько светодиодов, то их числа складываются. Таким образом, число 0 соответствует выключенным светодиодом, а 255 (сумма всех значений) всем включенным.

Таблица 6

№ светодиода	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Число, для включения	1	2	4	8	16	32	64	128

Для отправки информации на видеокамеру необходимо использовать команду

```
SHIFTOUT [dPin], [cPin], [Mode], [[Data]]
```

`[dPin]` – порт передачи данных,

`[cPin]` – порт синхронизации,

`[Mode]` – режим передачи синхросигнала и данных,

`[Data]` – данные.

Так будет выглядеть команда для включения двух крайних светодиодов:

```
SHIFTOUT 4, 8, LSBFIRST, [129]
```

Работа с самой видеокамерой отличается от работы с другими датчиками главным образом потому, что камера не только определяет наличие объекта, но так же может выдать его характеристики. Используемая в работе видеокамера может определять цвет и размеры объекта. Ограничением является то, что она работает только с объектами, которые имеют четко выраженный цвет, т.е. она не может работать с разноцветными объектами.

Передача и прием данных осуществляются с помощью команд `SERIN` и `SEROUT`.

```
SERIN [dPin], [Speed], [[Var]]
SEROUT [dPin], [Speed], [[Data]]
```

`[dPin]` – порт получения/передачи данных,

`[Speed]` – скорость передачи,

`[Var]` – переменная, куда будет записана полученные данные,

`[Data]` – передаваемые данные.

Перед работой с камерой необходимо сделать два обязательных действия:

- Настроить камеру на текущее освещение,
- Указать цвет объекта, который будет отслеживаться.

После калибровки камеры можно получать информацию об объекте. Если камера видит объект, она выдаст его координаты. Получая такую информацию в течение некоторого времени, можно узнать, подвижен или неподвижен объект. Процесс снятия информации с камеры выглядит следующим образом:

```
SEROUT 7, 84, ["TC",CR]
SERIN  9, 84, [STR RcvData\10]
```

Первой командой мы посылаем запрос Track color на камеру, второй, получаем информацию. Полученный массив `RcvData` будет состоять из десяти элементов, каждый из которых будет содержать следующую информацию:

0 – всегда 255,

1 – всегда код символа M,

2 – координата X середины объекта,
3 – координата Y середины объекта,
4 – координата X левого верхнего угла,
5 – координата Y левого верхнего угла,
6 – координата X правого нижнего угла,
7 – координата Y правого нижнего угла,
8 – размер объекта в пикселях,
9 – похожесть цвета объекта, на обученный цвет.

Порядок выполнения работы.

1. Загрузить программу lab6.bs2 в микроконтроллер.
2. Выполнить калибровку на текущее освещение. Функция №1, выбирается кнопкой “Select”.
3. Запомнить цвет отслеживаемого предмета в память. Функция №2. Для этого поднести предмет к объективу камеры на расстояние 2-3 см, нажать кнопку “Go” и дождаться звукового сигнала.
4. Опробовать возможность слежения за объектом. Функция №3.

Коммуникационные устройства

«Использование беспроводных средств коммуникации»

Общие сведения

Bluetooth — производственная спецификация беспроводных персональных сетей (англ. Wireless personal area network, WPAN).

Bluetooth обеспечивает обмен информацией между такими устройствами как карманные и обычные персональные компьютеры, мобильные телефоны, ноутбуки, принтеры, цифровые фотоаппараты, мышки, клавиатуры, джойстики, наушники, гарнитуры на надёжной, недорогой, повсеместно доступной радиочастоте для ближней связи.

Bluetooth позволяет этим устройствам сообщаться, когда они находятся в радиусе до 10-100 метров друг от друга (дальность очень сильно зависит от преград и помех), даже в разных помещениях.

Но технология Bluetooth - это намного больше, чем просто способ избавиться от проводов между существующими сегодня электронными устройствами. Эта технология дает возможность поднять эти устройства на новый уровень производительности, а также разработать целый класс новейших устройств с широкими коммуникационными функциями.

Класс	Максимальная мощность, мВт	Максимальная мощность, дБм	Радиус действия (приблизительно), м
1	100	20	100
2	2,5	4	10
3	1	0	1

Принцип действия Bluetooth.

Радиосвязь Bluetooth осуществляется в ISM-диапазоне (англ. Industry, Science and Medicine), который используется в различных бытовых приборах и беспроводных сетях (свободный от лицензирования диапазон 2,4-2,4835 ГГц). Спектр сигнала формируется по методу FHSS (Frequency Hopping Spread Spectrum — псевдослучайная перестройка рабочей частоты). Метод FHSS прост в реализации, обеспечивает устойчивость к широкополосным помехам, а оборудование стоит недорого.

Согласно алгоритму FHSS, в Bluetooth несущая частота сигнала скачкообразно меняется 1600 раз в секунду (всего выделяется 79 рабочих частот шириной в 1 МГц, а в Японии, Франции и Испании полоса у́же — 23 частотных канала). Последовательность переключения между частотами для каждого соединения является псевдослучайной и известна только передатчику и приёмнику, которые каждые 625 мкс (один временной слот) синхронно перестраиваются с одной несущей частоты на другую. Таким образом, если рядом работают несколько пар приёмник-передатчик, то они не мешают друг другу. Этот алгоритм является также составной частью системы защиты конфиденциальности передаваемой информации: переход происходит по псевдослучайному алгоритму и определяется отдельно для каждого соединения. При передаче цифровых данных и аудиосигнала (64 Кбит/с в обоих направлениях) используются различные схемы кодирования: аудио-сигнал не повторяется (как правило), а цифровые данные в случае утери пакета информации будут переданы повторно. Без помехоустойчивого кодирования это обеспечивает передачу данных со скоростями 723,2 Кбит/с с обратным каналом 57,6 Кбит/с, или 433,9 Кбит/с в обоих направлениях.

Совместимость.

Устройства, использующие технологию Bluetooth, могут быть функционально совместимы с одним или несколькими типами устройств. Два устройства могут быть связаны посредством Bluetooth – для этого они должны совместно использовать по крайней мере один общий профиль. Наладонный компьютер (Pocker PC) может обменивался данными с устройством EmbeddedBlue, для этого они оба должны поддерживают один и тот же профиль. Устройство EmbeddedBlue поддерживает профиль последовательного порта (SPP - Serial Port Profile), который является одним из самых ранних и наиболее широко поддерживаемых профилей.

Лабораторная работа 7.1
«Отправка данных роботу с управляющего компьютера»

Цель работы: обучение использованию беспроводных коммуникационных модулей для передачи данных и команд с управляющего компьютера на удаленные устройства.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Шаговый двигатель – 2 шт.;
- Модуль беспроводной связи EmbeddedBlue eb500-SER;
- Компьютер с модулем Bluetooth.

Теоретические сведения:

Режимы eb500-SER

eb500-SER поддерживает два основных операционных режима: командный и передачи данных. Как только подается питание, eb500-SER входит в командные режим и готов принимать последовательные команды. Заводские установки соединения: 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, and No Flow Control. eb500-SER поддерживает команду, чтобы изменить скорость бодов.

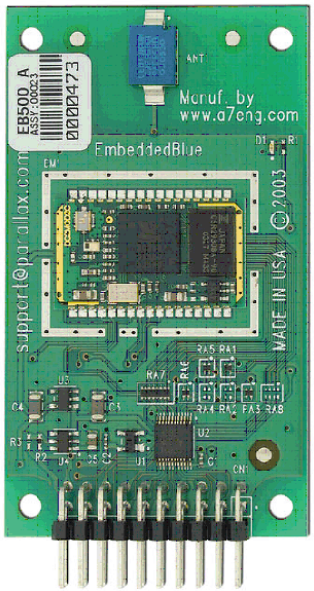


Рис. 7.1 Внешний вид EmbeddedBlue eb500-SER

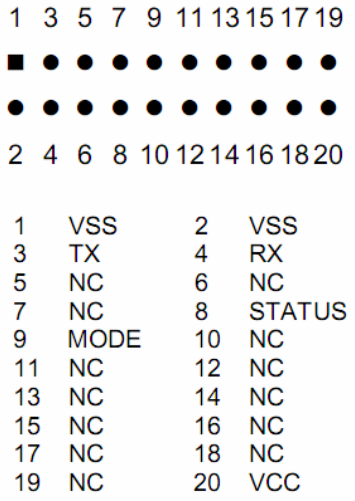


Рис. 7.2 Назначение контактов EmbeddedBlue eb500-SER

Как только eb500-SER подключен к другому устройству Bluetooth, он автоматически переключается в режим Данных. Все данные, переданные в этом режиме, будут отправлены удаленному устройству, поэтому никакие дальнейшие команды не могут быть отправлены, пока eb500-SER не будет отключен или переведен в Командный режим при помощи управления линии ввода/вывода или переключен в Командный режим по очереди.

Так как модуль, использующийся в данной работе поддерживает Serial Port Profile, то для связи с компьютером нужно создать виртуальный COM порт. Для ввода команд, для получения ответов от робота необходима любая терминальная программа. Например, HyperTerminal из стандартного пакета Windows.

Рис. 7.3 Вид установленного на плате Bluetooth модуля

Порядок создания подключения:

1. Открыть **HyperTerminal**
2. Дать название подключению (например eb500)
3. Выбрать подключение через **Serial Port**
4. В поле **Bits per** second выбрать **8**
5. В поле **Parity** выбрать **None**
6. В поле **Stop bits** выбрать **1**
7. В поле **Flow control** выбрать **None**
8. Нажать **OK**.

Принимать и передавать данные с помощью этого модуля очень просто.

Для получения данных модулем используется команда

```
SERIN 0,84,[Data]
```

Для передачи данных:

```
SEROUT 1,84,["Data"]
```

Пример программы:

```
Data VAR Byte(20)
PAUSE 1000
Main:
  SERIN 0,84,[STR Data\20\CR]
  DEBUG STR Data,CR
  GOTO Main
```

Порядок выполнения работы:

1. Написать программу получения данных модулем и вывода их в окно отладчика;
2. Передать произвольные последовательности символов;

3. Написать программу беспроводного управления движением робота;

Рекомендации:

Используйте команду `BRANCH CmdData,[#1, #2, #3, #4]`

```
CmdData – полученная информация от модуля,
#1–4 – имя подпрограммы на выполнение.
Main:
  SERIN 0,84,[DEC1 CmdData]
  BRANCH CmdData,[#1, #2, #3, #4]
  GOTO Main
```

Лабораторная работа 7.2

«Управление роботом через при помощи устройства на базе ОС Android»

Цель работы: обучение использованию беспроводных коммуникационных модулей для передачи данных и команд с мобильного устройства на удаленные устройства.

Приборы и принадлежности:

- Board of Education с микроконтроллером BS2;
- Сервопривод – 2 шт.;
- Модуль беспроводной связи EmbeddedBlue eb500-SER или RBT-001;
- Устройство с ОС Android и модулем Bluetooth.

Теоретические сведения:

Режимы RBT-001

Также как и eb500-SER, RBT-001 поддерживает два основных операционных режима: командный и передачи данных. Как только подается питание, модуль входит в командные режим и готов принимать последовательные команды. Заводские установки соединения: 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, and No Flow Control. Есть возможность менять скорость бодов.

Основное отличие данного модуля от описанного в предыдущей лабораторной работе в нумерации контактов. Подключение к плате робота осуществляется через UART-адаптер. Протокол UART (Universal asynchronous receiver/transmitter) или, по-русски, УАПП (универсальный асинхронный приемопередатчик) — старейший и самый распространенный на сегодняшний день физический протокол передачи данных. В данном адаптере используется наиболее известный из семейства UART протокол RS-232 или же COM-port.

Передача данных в UART осуществляется по одному биту в равные промежутки времени. Этот временной промежуток определяется заданной **скоростью UART** и для конкретного соединения указывается в бодах (битах в секунду). Существует общепринятый ряд стандартных скоростей: 300 бод, 600 бод, 1200 бод, 2400 бод, 4800 бод, 9600 бод, 19200 бод, 38400 бод, 57600 бод, 115200 бод, 230400 бод, 460800 бод, 921600 бод.

Помимо собственно информационного потока UART автоматически вставляет в поток синхронизирующие метки, так называемые **стартовый и стоповый биты**. При приеме эти лишние биты удаляются из потока. Обычно стартовый и стоповый биты обрамляют один байт информации (8 бит), однако встречаются реализации UART которые позволяют передавать по 5,6,7, 8 или 9 бит. Обрамленные стартом и стопом биты являются минимальной посылкой.

Рис. 7.4 Внешний вид RBT-001

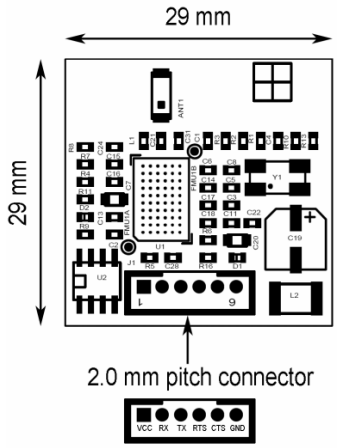


Рис. 7.5 Назначение контактов RBT

Для соединения мобильного устройства и робота будем использовать программу Cellbots. Данная программа предназначена для управления роботами на основе различных контроллеров. Она подходит как и для установки соединения, так и непосредственно для управления роботом.

Порядок создания подключения:

1. Открыть **Cellbots,** инициировать добавление нового устройства.
2. Выбрать тип устройства **Default controller**
3. Провести поиск устройств
4. Подключиться к устройству, введя код подтверждения (0000)
5. Нажать **Done**.

Данный контроллер, как и любой участник Bluetooth сети, обладает MAC-адресом – уникальным идентификатором устройства, который также отображается на экране с профилем устройства.

Принимать и передавать данные с помощью этого модуля очень просто.

Для получения данных модулем используется команда

```
SERIN 2,84,[Data]
```

Для передачи данных:

```
SEROUT 0,84,["Data"]
```

Фрагмент программы:

```
direction VAR Word
x VAR Word
y VAR Word
RX CON 2
TX CON 0
Baud CON 84
main:
DO
SERIN RX, Baud, 20, No_Data, [direction]
SERIN TX, Baud, 20, No_Data, [dat]
No_Data:
y = 0
DEBUG direction
IF direction = %01100110 THEN GOSUB forward
```

В данном фрагменте инициализируется соединение между модулем и мобильным устройством и даётся ссылка на выполнение подпрограммы при получении от устройства команды, соответствующей нажатию кнопки вперёд.

При нажатии на другие управляющие кнопки Cellbots посылает следующие команды: вправо 01110010, влево 01101100, назад 01100010, стоп 01110011.

Порядок выполнения работы:

1. Соединить модуль и мобильное устройство посредством Bluetooth;
2. Оптимизировать программу в примере и написать на её основе программу беспроводного управления движением робота;
3. Написать программу беспроводного управления движением робота для езды по полю, состоящему из клеток 16x16 см с плавным стартом.

Контрольные вопросы к лабораторным работам

Лабораторная работа №1.1

1. Почему установленные сервоприводы вращаются в разные стороны при одинаковых импульсах, подаваемых на двигатели?
2. Какие величины импульсов необходимы для максимальной скорости сервопривода?
3. Рассчитайте, какое количество повторений цикла необходимо для вращения сервопривода в течении 7.5 секунд.
4. Каким способом управляются данные сервоприводы?

Лабораторная работа №1.2

1. Каким способом управляются данный сервопривод?
2. Какая величина импульсов необходима для положения в 45°?
3. Для каких целей может применяться данная модель сервопривода?

Лабораторная работа №2.1

1. Для каких целей применяется инфракрасное излучение?
2. На какое количество диапазонов делят инфракрасное излучение?
3. Какие тела могут излучать в инфракрасном спектре?
4. Какая длина волны у инфракрасного диода, применяемого в данной лабораторной работе?

Лабораторная работа №2.2

1. Какой принцип используется в паре излучатель-приемник в данной лабораторной работе?
2. Для каких целей возможно применять ИК датчики в подвижных роботах?
3. В каком случае целесообразно применение ИК датчиков?

Лабораторная работа №3.1

1. Что называется ультразвуковым излучением?
2. Для каких целей возможно применять УЗ датчики в подвижных роботах?
3. Принцип действия УЗ датчика.
4. Конструкция УЗ датчика.

Лабораторная работа №3.2

1. Каков диапазон измерений УЗ датчика, используемого в данной лабораторной работе?
2. Каково время отражения сигнала от объекта на расстоянии 50см?
3. Для чего применяется поворотное устройство в паре с УЗ датчиком?

4. Какие задачи может решать колесный робот, оснащенный лишь УЗ датчиком и поворотным устройством?

Лабораторная работа №3.3

1. В каких случаях УЗ датчик может быть неэффективен?
2. Какие способы повышения эффективности УЗ датчика можно применить?
3. Какова скорость звука в воздухе при комнатной температуре?

Лабораторная работа №4.

1. Для каких целей целесообразно применять электронный компас на подвижном роботе?
2. Объясните возможные причины появления шумов измерения.
3. Способы повышения точности и конкретизации значений электронного компаса.
4. Поясните, почему компас, использующийся в данной лабораторной работе, является двухосевым?

Лабораторная работа №5.1-5.2

1. Для каких целей целесообразно применять энкодеры?
2. Сколько импульсов энкодера соответствует полному обороту колеса?
3. Опишите принцип действия энкодера, применяемого в данной лабораторной работе.
4. Существуют ли энкодеры с другими типами действия?
5. Как повысить точность энкодера?

Лабораторная работа №6.

1. Что такое «Датчики изображения» и «Системы технического зрения»?
2. Положительные и отрицательные стороны систем технического зрения.
3. Можем ли мы, используя видеокамеру в данной работе, получать от нее актуальную информацию с частотой 30 раз в минуту?
4. Какие параметры влияют на качество данных, получаемых от видеокамеры?

Лабораторная работа №7.1

1. В каких устройствах используется связь Bluetooth?
2. В каком частотном диапазоне работают устройства Bluetooth?
3. Каковы методы защиты конфиденциальности передаваемой информации?

4. Какова максимальная скорость передачи данных по каналу Bluetooth?
5. Как обеспечивается совместимость различных устройств Bluetooth?

Лабораторная работа №7.2

1. Что такое MAC-адрес? Какой MAC-адрес у используемого Bluetooth-модуля?
2. Что такое UART(УАПП)?
3. Какие стандартные скорости бывают у этого интерфейса?

Какое количество информации обычно обрамляют стартовый и стоповый биты? Каков размер минимальной посылки через UART?