Сэру Патрику Стюарту (Patrick Stewart) – за то, что он был
ярким светом человечества даже в самые мрачные дни.
И моему любимому псу Луке (Luca) – за молчаливое признание того,
что некоторые прогулки были короче, чем обычно,
во время написания этой книги.

– Тобиас Тил (Tobias Theel)

Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге, – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте **www.dmkpress.com**, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства ДМК Пресс и Packt Publishing очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты **dmkpress@gmail.com**.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Тобиас Тил – технический руководитель и разработчик в немецком стартапе FinTech fino, а также ведущий инженер-программист в стартапе RegTech и ClariLab. Является архитектором программного обеспечения и экспертом в языках Go, TinyGo и параллельно в C# и Java. Прошел сертификацию по iSAQB.

Тил является активным участником сообщества на StackOverflow и входит в число 10 % самых ведущих его представителей по C# и Unity3D и 20 % – по .NET, Go и Visual Studio.

В свободное от работы время его можно найти разрабатывающим игры, в основном на Game Jame, таких как Ludum Dare Jam. Задача данных соревнований – это разработать игру с нуля в течении 72 ч. Будучи активным спикером на технических дискуссиях и участником многочисленных хакатонов, Тил любит делиться своими знаниями в области разработки программного обеспечения с другими энтузиастами.

Я хочу поблагодарить свою команду в ClariLab за то, что она предоставила мне так много выходных дней, чтобы закончить эту книгу. Также выражаю особую благодарность Йоханнесу Колате (Johannes Kolata) за столь ценный вклад в подготовку книги.

О рецензентах

Энрико фон Отте (Enrico von Otte) начал изучать программирование в 11-летнем возрасте на старом добром Commodore C64. Позже он кодировал на Amiga 2000, а в середине 90-х перешел на персональный компьютер. Он работает в области профессиональной разработки программного обеспечения с 2005 года. После длительного опыта разработки аппаратного обеспечения, близкого к программному обеспечению на C и C++, он перешел в мир C# в 2008 году.

До 2015 года Энрико разрабатывал GIS-системы и системы управления документами. Теперь он профессиональный архитектор программного обеспечения с сильной привязанностью к self-made-коду и девизом «Сборка не разработка».

Йоханнес Колата (Johannes Kolata) начал программировать 8 лет назад, работая над системами управления документами. С 2018 года Колата работает инженером по программному обеспечению в немецкой компании FinTech fino, которая наиболее известна созданием первого цифрового коммутатора банковских счетов. Попутно он стал экспертом в разработке на C#, Java, Golang и C++, а также получил представление о множестве других языков программирования. Помимо работы в индустрии FinTech, он страстный энтузиаст открытого исходного кода и 3D-печати и с нуля создает системы домашней автоматизации, которые характеризуются взаимодействием CAN (Controller Area Network), затрагивающим API (Application Programming Interface) и панель инструментов Angular. Когда Колата не работает над проектами, он участвует в Game Jame и хакатонах.

Предисловие

Если языки JavaScript или C# могут применяться для программирования микроконтроллеров, то Go может сделать это еще лучше. В то время как стандартный Go производит огромные бинарные коды, TinyGo производит двоичные коды, которые подходят для самых маленьких устройств. Почему же вы должны выбрать Go для микроконтроллера и программирование Wasm (сокращение от WebAssembly)? Мои любимые причины в том, что язык Go легко выучить, легко читать и легко на нем писать. Кроме того, Go поставляется с мощной стандартной библиотекой, которая легко привязывается и имеет широкие возможности параллелизма.

Если вы любите язык программирования Go, то эта книга для вас. После прочтения у вас будут все инструменты и знания, необходимые для создания всех проектов, связанных с микроконтроллерами, о которых вы когда-либо мечтали. Кроме того, в качестве дополнительного преимущества вы сможете создавать информационные панели и приложения для управления домом с помощью Wasm для ваших проектов домашней автоматизации. Всего этого можно достичь с помощью TinyGo.

Если вы никогда раньше не работали с микроконтроллерами, вот несколько причин, по которым это следует попробовать:

- если вы уже являетесь программистом, здорово видеть, как ваш код влияет на работу реальных устройств. Это действительно прекрасное чувство – завершить проект и, наконец, увидеть, как вращается двигатель, мигает светодиод, раздается звуковой сигнал и т. д.;
- вы будете постоянно узнавать что-то новое и глубже понимать, как работают компьютеры в целом, поскольку познакомитесь с различными типами шин компьютерных систем, протоколов, аппаратных интерфейсов и многим другим;
- возможности практически безграничны, когда вы играете с микроконтроллерами. Вы не привязаны к тому, что доступно на рынке, так как можете просто построить все самостоятельно;
- можно научиться писать небольшие эффективные программы, чтобы сообщить микроконтроллеру, чего вы от него хотите. В целом это также поможет вам улучшить навыки разработчика;
- вы можете вносить свой вклад в интересные проекты и вступать в контакт с отличными сообществами единомышленников.

Для кого предназначена книга

Если вы разработчик Go, который хочет программировать маломощные устройства и оборудование, такие как Arduino UNO и Arduino Nano

IoT 33, или который хочет расширить свои знания об использовании Go с WebAssembly для программирования на данном языке в браузере, тогда эта книга для вас. Программисты-любители языка Go, которые заинтересованы в том, чтобы узнать больше о TinyGo, работая над проектами DIY, также найдут это практическое руководство полезным.

О чем эта книга

Глава 1, «Начало работы с TinyGo», показывает, как настроить TinyGo и скомпилировать свою первую программу!

В главе 2, «Построение системы управления светофорами», вы создадите систему управления светофорами, включая световые индикаторы для пешеходов и кнопку управления; кроме этого, вы изучите, как использовать функции Gorutine (горутины) в TinyGo.

В главе 3, «Создание кодового замка с использованием клавиатуры», рассматривается использование клавиатуры 4×4 и сервомотора для создания замка, который открывается при вводе правильного пароля.

Глава 4, «Создание системы полива растений», объясняет, как использовать различные типы датчиков, чтобы построить автоматическую систему полива растений. Вам больше не придется поливать растения вручную!

Глава 5, «Создание таймера для безконтактного мытья рук», посвящена использованию четырехзначного семисегментного дисплея и ультразвукового датчика расстояния для распознавания движения недалеко расположенного объекта, чтобы запустить таймер, который будет регулировать длительность процесса мытья рук.

Глава 6, «Подключение дисплеев для связи с использованием интерфейсов I2C и SPI», объясняет концепции **межинтегрированной схемы** (I2C) и **последовательного периферийного интерфейса** (SPI) на примере использования дисплеев, которые взаимодействуют с помощью шин I2C и SPI. К концу главы вы узнаете, как использовать различные типы дисплеев в TinyGo.

Глава 7, «Мониторинг погоды на панели управления Wasm TinyGo», посвящена созданию и обслуживанию приложения Wasm, которое отображает данные датчиков, отправленные с Arduino Nano 33 IoT по Wi-Fi.

Глава 8, «Автоматизация и мониторинг вашего дома с помощью панели управления TinyGo Wasm», объясняет, как управлять и контролировать устройства в вашем доме с помощью панели управления Wasm.

Максимальная отдача от книги

Все примеры кода были протестированы с Go 1.16.2 в Ubuntu, но они также будут работать с будущими выпусками Go и в других операционных системах. Код на Visual Studio использовался в качестве редактора на протяжении всей книги, но можно использовать любой другой редактор.

Программное и аппаратное обеспечение, описанное в книге	Требования к операционной системе
Go 1.15.x или новее	Windows, macOS или Linux
Visual Studio Code	Windows, macOS или Linux

Если вы используете цифровую версию этой книги, мы советуем вам писать код самостоятельно или получить доступ к коду через репозиторий GitHub (ссылка доступна в следующем разделе). Это поможет избежать любых потенциальных ошибок, связанных с копированием и вставкой кода.

Я бы хотел увидеть проекты, которые вы создадите после прочтения этой книги в социальных сетях. Не стесняйтесь отмечать меня в Twitter, используя следующий тег: **@Nooby_Games**.

Загрузите файлы с примерами кода

Вы можете скачать примеры файлов кода для этой книги с GitHub по адресу **https://github.com/PacktPublishing/Creative-DIY-Microcontroller-Projects-with-TinyGo-and-WebAssembly**. В случае обновления кода он будет обновлен в существующем репозитории GitHub.

У нас также есть другие пакеты с кодами из нашего богатого каталога книг и видео, доступных по адресу **https://github.com/PacktPublishing/**. Посмотрите их!

Код в действии

Видеоролики с рабочими кодами для этой книги можно посмотреть по адресу **https://bit.ly/3cYZOh4**.

Загрузите цветные изображения

Мы также предоставляем PDF-файл с цветными изображениями скриншотов/диаграмм, используемых в этой книге. Вы можете скачать их здесь: **https://static.packt-cdn.com/downloads/9781800560208_ColorImages.pdf**.

Используемые соглашения

В этой книге используется ряд текстовых соглашений. Код в тексте указывает кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, фиктивные URL-адреса, вводимые пользователем, и дескрипторы Twitter. Вот пример: «Когда мы получаем начало команды, мы добавляем все последующие символы в *commandBuffer*».

Блок кода задается следующим образом:

```
data, err := uart.ReadByte()
if err != nil {
```

```
    println(err.Error())
}
```

Когда мы хотим привлечь ваше внимание к определенной части блока кода, соответствующие строки или элементы выделены жирным шрифтом:

```
func main() {
    blocker := make(chan bool, 1)
    <-blocker
    println("это никогда не печатается")
}
```

Любой ввод или вывод командной строки записывается следующим образом:

```
tinygo flash -target=arduino-nano33 Chapter06/tinygame/main.go
```

Жирный шрифт указывает на новый термин, важное слово или слова, которые вы видите на экране. Например, слова в меню или диалоговых окнах отображаются в тексте следующим образом. Пример: «Значение довольно стабильно на уровне **37888**».

> **Советы или важные примечания**
>
> Выглядят так.

Обратная связь

Отзывы наших читателей всегда приветствуются.

Общая обратная связь: если у вас есть вопросы по какому-либо аспекту этой книги, укажите название книги в теме своего сообщения и напишите нам по адресу **customercare@packtpub.com**.

Ошибки: хотя мы позаботились о том, чтобы обеспечить точность нашего контента, ошибки все же случаются. При обнаружении ошибки в этой книге мы были бы признательны, если бы вы сообщили нам об этом. Пожалуйста, посетите **www.packtpub.com/support/errata**, выберите свою книгу, нажмите на ссылку «форма отправки с ошибками» и введите данные.

Пиратство: если вы столкнетесь с любыми незаконными копиями наших работ в любой форме в интернете, мы будем признательны, если вы сообщите нам адрес местонахождения или название веб-сайта. Пожалуйста, свяжитесь с нами по адресу **copyright@packt.com** со ссылкой на материал.

Если вы заинтересованы в том, чтобы стать автором: при наличии темы, в которой вы разбираетесь, и, если вы заинтересованы в написании или участии в книге, пожалуйста, посетите **https://authors.packtpub.com/**.

Отзывы

Пожалуйста, оставьте отзыв. После того как вы прочитали и использовали эту книгу, почему бы не оставить отзыв на сайте, на котором вы ее купили? Потенциальные читатели смогут увидеть и использовать ваше непредвзятое мнение для принятия решений о покупке. Мы в Packt сможем понять, что вы думаете о наших продуктах, а наши авторы смогут увидеть ваши отзывы о своей книге. Спасибо!

Для получения дополнительной информации о Packt, пожалуйста, посетите **https://www.packtpub.com/**.

Глава 1

Начало работы с TinyGo

На мой взгляд, язык Go легко выучить, легко читать и легко, писать. Язык не перегружен причудливыми функциями, а скорее ориентирован на лаконичность. Встроенный параллелизм, быстрое время компиляции, высокая производительность выполнения и богатые стандартные библиотеки создают отличное сочетание для этого потрясающего языка. Вот почему я хочу отправиться с вами в путешествие от очень простых высокоуровневых программ Go до глубин микроконтроллеров, использующих всю мощь TinyGo.

В этой главе мы собираемся настроить TinyGo и узнать, как заставить наш готовый код работать в VS Code и в других редакторах. После того как это будет сделано, мы рассмотрим плату Arduino UNO и его технические характеристики. Затем необходимо сравнить TinyGo с Go и поговорить о том, что делает TinyGo особенным по сравнению с другими языками для микроконтроллеров. В конце этой главы мы напишем, скомпилируем, развернем и запустим нашу первую программу на языке TinyGo для реального микроконтроллера. Рассмотрев все эти темы, вы узнаете, как писать, создавать и запускать программы на микроконтроллерах.

В этой главе рассмотрим следующие основные темы:

- знакомимство с языком TinyGo;
- настройку TinyGo;
- настройку интеграции IDE с TinyGo;
- Arduino UNO;
- тестирование устройств программой Hello World.

Технические требования

Для того чтобы продолжить, вам необходимо иметь следующее:

- Go должен быть установлен;
- GOPATH должен быть настроен;
- Git должен быть установлен;
- Arduino Uno – предпочтительно версия Rev3, но можно использовать другие Arduino-платы.

Вы можете найти все примеры кода из этой главы в следующем репозитории GitHub: **https://github.com/PacktPublishing/Creative-DIYMicrocontroller-Projects-with-TinyGo-and-WebAssembly/tree/master/Chapter01**. Видео с рабочим кодом для этой главы можно найти здесь: **https://bit.ly/3mLFCCJ**.

Знакомимся с языком TinyGo

TinyGo – это отдельно написанный компилятор со своей собственной реализацией среды выполнения. Он предназначен для программирования микроконтроллеров, веб-сборки (WASM) и инструментов CLI. TinyGo активно использует инфраструктуру LLVM для оптимизации и компиляции бинарного кода, понятного микроконтроллеру.

Первый релиз TinyGo (v0.1) был опубликован 1 февраля 2019 года на GitHub. С тех пор проект быстро внедрил множество функций и никогда не прекращал добавлять поддержку большего количества микроконтроллеров, датчиков, дисплеев и других устройств.

2 февраля 2020 года TinyGo объявил, что теперь он официально является проектом, спонсируемым Google. Это был большой шаг в развитии проекта.

Как работает TinyGo

Компилятор TinyGo использует набор шагов, отличный от других языков, для преобразования исходного кода Go в машинный код. Однако мы не будем вдаваться в подробности, но давайте взглянем на усеченную структуру компилятора.

1. Мы пишем исходный код на Go.
2. Этот исходный код переводится в Go SSA (**Static Single Assignment**).
3. SSA Go преобразуется в LLVM IR (Low Level Virtual Machine) с помощью пакета компилятора TinyGo.
4. Код инициализации в LLVRM IR интерпретируется пакетами взаимодействия TinyGo. Этот шаг оптимизирует глобальные значения, константы и многое другое.
5. Затем результат оптимизируется с помощью некоторых проходов оптимизации LLVM (например, оптимизации *string* в *[] byte*).
6. Затем результат снова оптимизируется модификатором LLVM.
7. Далее, некоторые исправления выполняются пакетом компилятора.
8. И в качестве последнего шага LLVM создает машинный код.

Если сейчас это звучит сложно, не волнуйтесь – нам не нужно заботиться об этом процессе. TinyGo делает все это за нас. Теперь давайте посмотрим, что делает TinyGo особенным по сравнению с Go.

Сравнение TinyGo с Go

TinyGo может скомпилировать некоторые, но не все программы Go. Давайте рассмотрим пример, который может быть скомпилирован обеими языковыми оболочками. Для примера напишем небольшую программу Hello World в Go – создадим ее и посмотрим размер программы.

1. Это самая минимальная программа Hello World, которую я могу придумать на данный момент:

```
package main
func main() {
print("Hello World\n")
}
```

   Для печати строки не требуется внешний пакет, такой как `fmt`.

2. Я буду использовать Go 1.15.2 в операционной системе Ubuntu 20.01. Чтобы проверить текущую установленную версию Go, используйте команду *go version*:

```
$ go version
go version go1.15.2 linux/amd64
```

3. Мы создаем файл с программой с помощью команды *go build*:

```
$ go build ./ch1/hello-world/
```

4. Теперь проверяем размер с помощью команды *ls –l*:

```
$ ls -l
-rwxrwxr-x 1 tobias tobias 1231780 Okt 4 19:31 hello-world
```

Итак, программа имеет 1 231 780 байт, что составляет 1,23178 Мб. Это довольно много для программы, состоящей всего из четырех строк кода.

> **Примечание**
>
> Команда *ls* доступна не во всех операционных системах. Если вы хотите самостоятельно проверить размеры, необходимо использовать инструменты, доступные в вашей операционной системе. Размер бинарного файла может отличаться при его анализе, так как команда разработчиков языка Go продолжает оптимизировать компилятор. Кроме того, размер бинарного файла может отличаться при сборке для других операционных систем.

Теперь давайте проверим, каков размер той же программы, но на этот раз скомпилированной с использованием TinyGo. Поскольку TinyGo не

поддерживает создание двоичного кода для Windows, я позабочусь о компиляции, поэтому вы можете просто сравнить размеры, изложенные в книге.

1. Я использовал следующую команду для создания бинарного файла:

```
$ tinygo build -o hello-world-tiny ch1/hello-world/main.go
```

   Команда *tinygo build* имеет синтаксис, аналогичный команде *Go build*.

2. Затем я проверил размер с помощью команды *ls –l*, как мы делали раньше:

```
$ ls -l
-rwxrwxr-x 1 tobias tobias 1231780 Okt 4 19:31 hello-world
-rwxrwxr-x 1 tobias tobias 21152 Okt 4 19:39 hello-world-tiny
```

Мы видим, что версия TinyGo нашей программы Hello World составляет лишь малую часть того размера, который был выдан компилятором Go. Версия TinyGo составляет всего 21 152 байта, что примерно 0,021152 Мб. Размер программа на TinyGo в 58 раз меньше программы, написанной на языке Go. Это огромная разница. Если вы все еще хотите проверить это самостоятельно, то можете сделать это после настройки TinyGo.

Мы узнали с вами, что TinyGo может компилировать некоторые, но не все программы Go. Кроме того, также узнали, что программы, скомпилированные с помощью TinyGo, очень малы. В следующих разделах мы узнаем, почему TinyGo не может скомпилировать все программы Go и какие функции TinyGo предлагает, а Go не предлагает.

Поддерживаемые языковые функции

TinyGo поддерживает часть функций языка Go, но не все поддерживается прямо сейчас. Горутины (goroutines) и каналы работают на большинстве микроконтроллеров. Рефлексия поддерживается для большинства типов. Хотя срезы (slice) и поддерживаются, но при работе с картами (хеш-таблицами) могут возникнуть некоторые проблемы. Поддерживаются только определенные типы данных: строки, целые числа, указатели, структуры и массивы. Таким образом, в целом значительная часть языка Go поддерживается в TinyGo.

Поддерживаемые стандартные пакеты

Большая часть стандартной библиотеки также поддерживается в TinyGo. Однако на момент написания статьи большинство *net*- и *crypto*-пакетов все еще не компилируется. Это означает, что если вы их будете импортировать, то, скорее всего, получите ошибки компиляции. Вы можете просмотреть список поддерживаемых в настоящее время стандартных пакетов здесь: **https://tinygo.org/lang-support/stdlib/**.

> **Примечание**
>
> Каждая функция в пакете, указанном в таблице поддержки, в действительности может не использоваться в TinyGo. Некоторые функции все еще могут вызывать ошибки компиляции.

Операции volatile

Операции volatile могут использоваться для чтения и записи из регистров, отображенных в памяти. Значения внутри этих регистров могут меняться между несколькими считываниями без ведома компилятора. Компилятор не знает о последствиях этих операций, поэтому они называются изменчивыми (volatile).

У Go нет оператора volatile, поэтому TinyGo предоставляет пакет volatile. В большинстве случаев нам не понадобятся операции volatile, так как они абстрагируются машинным пакетом.

Встроенный ассемблер

Язык ассемблера (ASM) – это язык, специально разработанный для определенной архитектуры процессора. Он необходим потому, что сборка зависит от набора инструкций машинного кода. Пакеты TinyGo, предназначенные для конкретных устройств, предоставляют пакеты такой сборки. Это позволяет нам использовать встроенный ассемблерный код в наших программах Go, что невозможно в стандартном языковом пакете Go.

Распределение памяти

Heap (куча) – это часть памяти, в которой во время выполнения происходят динамические распределения и освобождения. Поэтому, когда наше приложение хочет зарезервировать часть памяти, оно взаимодействует с heap, чтобы зарезервировать память. Затем эта часть памяти будет помечена как используемая. Поскольку это пространство довольно ограничено на микроконтроллерах, а сбор мусора является дорогостоящим и медленным, TinyGo пытается оптимизировать распределение памяти. В результате часто объекты могут быть распределены статически, а не динамически.

Сборка мусора

Сборка мусора – это процесс освобождения памяти. Поэтому, когда вашему приложению больше не нужна часть памяти, которую оно запрашивало ранее, эта память помечается как неиспользуемая (свободная).

Для этой цели TinyGo реализовал свой собственный вариант сбора мусора. TinyGo использует сборку мусора с консервативной меткой/разверткой, где консервативность означает, что сборщик мусора (GC) не знает, что является указателем, а что – нет. Процесс GC разделен на две части.

- **Отметка**: на этапе маркировки gc помечает объекты как доступные.
- **Очистка**: на этапе очистки gc освобождает память, помечая области, где отсутствуют объекты, как свободные. Эти освобожденные области памяти затем могут быть повторно использованы для выделения под новые объекты.

Теперь мы знаем, что такое TinyGo и какие существуют различия между TinyGo и Go. Также узнали, что такое куча, GC и пакет volatile. Следующий логический шаг –продолжить изучать и настраивать TinyGo, что мы и сделаем в следующем разделе.

Установка TinyGo

Самый простой способ установить TinyGo и все его зависимости – это выполнить быстрый запуск руководства для Linux, macOS, Windows и Docker по следующей ссылке: **https://tinygo.org/getting-started/**.

Поскольку эти руководства охватывают важные части, я расскажу только о части быстрого запуска для архитектур на базе x64 и только для операционных систем на базе Debian, таких как Ubuntu для Linux.

Первое, что нужно сделать, прежде чем мы начнем настройку, – это проверить последнюю версию TinyGo. Для этого перейдите на **https://github.com/tinygo-org/tinygo/releases** и узнайте последнюю версию выпуска. Теперь запишите эту информацию где-нибудь или запомните, так как мы будем использовать ее позже.

Установка в Linux

Следующие шаги охватывают установку TinyGo для разновидности операционной системы Linux, которая основана на Debian.

1. Мы используем следующую команду, чтобы загрузить пакет *deb* с GitHub и установить его с помощью *dpkg*:

   ```
   wget https://github.com/tinygo-org/tinygo/releases/
   download/v0.15.0/tinygo_0.15.0_amd64.deb
   sudo dpkg -i tinygo_0.15.0_amd64.deb
   ```

   Вы можете изменить версию, указанную в пути и имени файла, на самую новую версию выпуска, которую нашли ранее.

2. Теперь мы должны добавить TinyGo в GOPATH. Используйте следующую команду:

   ```
   export PATH=$PATH:/usr/local/tinygo/bin
   ```

   Вы также можете расширить GOPATH, отредактировав свой файл *.profile* или *.netrc*.

3. Следующим шагом является проверка установки. Используйте команду *tinygo version*, чтобы убедиться, что TinyGo успешно установлен: