Getting Started with Arduino and Go

Agus Kurniawan

Copyright

Getting Started with Arduino and Go

Agus Kurniawan

1st Edition, 2015

* Cover photo is credit to Fajar Ramadhany, Bataviasoft, http://bataviasoft.com/.

** Arduino logo is taken from http://www.arduino.cc/ .

Table of Contents

Preface

This book was written to help anyone want to get started with Arduino and Go. It describes the basic elements of the integration of Arduino and Go.

Agus Kurniawan

Depok, March 2015

1. Preparing Development Environment

1.1 Arduino

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. This board uses Atmel microcontroller series. There are many Arduino hardware models that you can use. Further information about Arduino products, you can visit on website http://arduino.cc/en/ .

You must one Arduino hardware to follow practices in this book. I recommend to obtain one of the following Arduino hardware:

- Arduino Uno
- Arduino Leonardo
- Arduino Mega 2560
- Arduino Due

You can buy this product on your local electronic store. You also can order it by online. Find it on http://arduino.cc/en/Main/Buy. The following is the list of Arduino store you can buy

- Arduino store, http://store.arduino.cc/
- Amazon, http://www.amazon.com
- Cooking-hacks, http://www.cooking-hacks.com/index.php/shop/arduino.html
- RS Components, http://www.rs-components.com
- Element 14, http://www.element14.com
- EXP-Tech, http://www.exp-tech.de

Because Arduino is an open-source hardware, people can build it. It's called Arduino compatible. Generally it's sold in low prices.

1.1.1 Arduino Uno

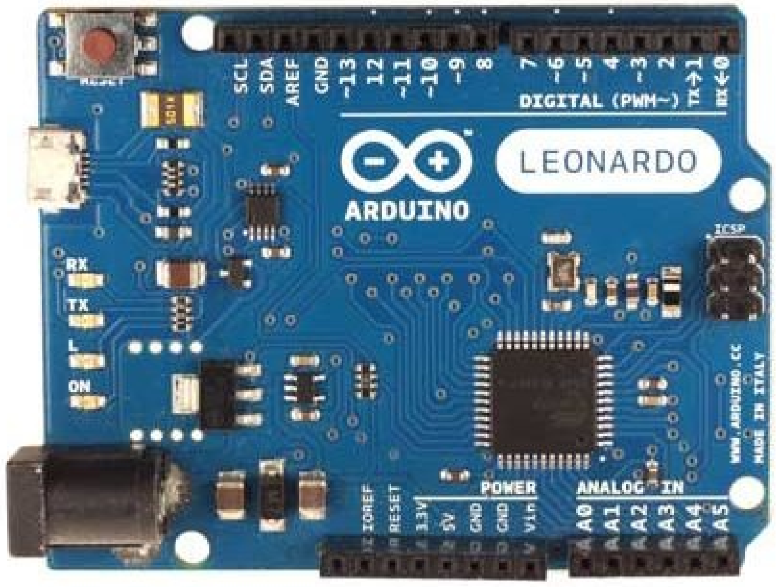
The Arduino Uno is a microcontroller board based on the ATmega328. You can download the datasheet file, http://www.atmel.com/dyn/resources/prod_documents/doc8161.pdf .

Further information about Arduino Uno, you can read it on http://arduino.cc/en/Main/ArduinoBoardUno .

1.1.2 Arduino Leonardo

The Arduino Leonardo is a microcontroller board based on the ATmega32u4. Download datasheet for this product on
http://www.atmel.com/dyn/resources/prod_documents/7766S.pdf .

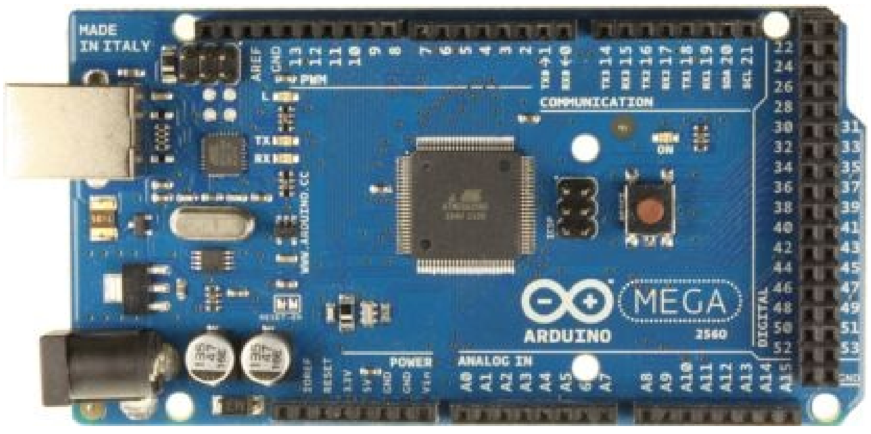
Visit this product to get the further information on
http://arduino.cc/en/Main/ArduinoBoardLeonardo .



1.1.3 Arduino Mega 2560

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. You can download the datasheet file on
http://www.atmel.com/dyn/resources/prod_documents/doc2549.PDF.

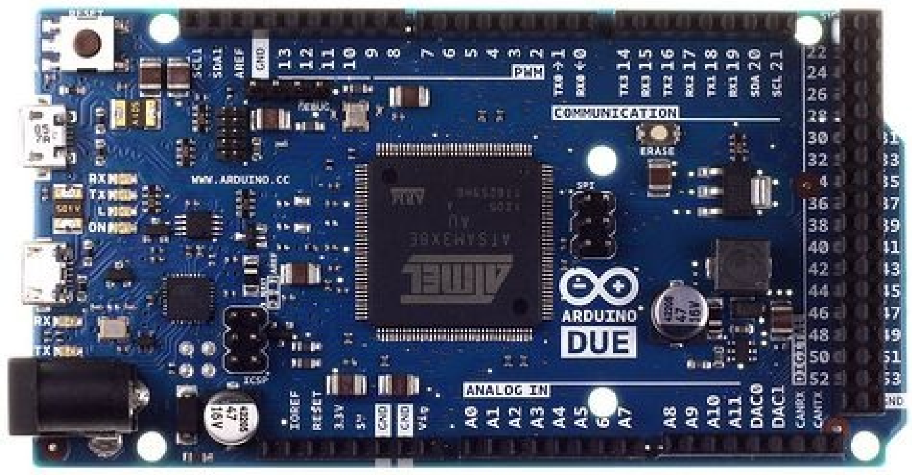
Further information about Arduino Mega 2560, you can visit on http://arduino.cc/en/Main/ArduinoBoardMega2560 .



1.1.4 Arduino Due

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. You can download the datasheet, http://www.atmel.com/Images/doc11057.pdf.

If you want to know about Arduino Due, I recommend to visit this website, http://arduino.cc/en/Main/ArduinoBoardDue.



1.2 Electronic Components

We need electronic components to build our testing, for instance, Resistor, LED, sensor devices and etc. I recommend you can buy electronic component kit.

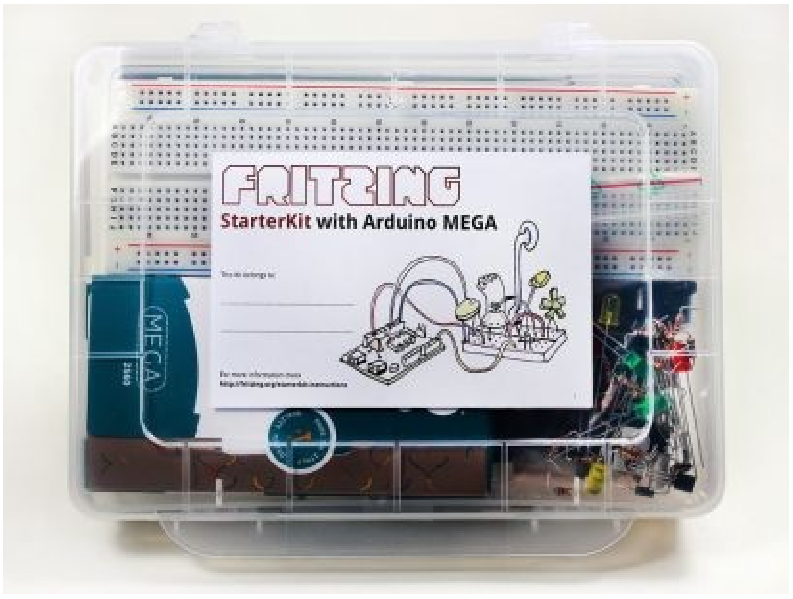
1.2.1 Arduino Starter Kit

Store website: http://arduino.cc/en/Main/ArduinoStarterKit

1.2.2 Fritzing

Store website: http://shop.fritzing.org/ .

You can buy Fritzing Starter Kit with Arduino UNO or Fritzing Starter Kit with Arduino Mega.



1.2.3 Cooking-Hacks: Arduino Starter Kit

Store website: http://www.cooking-hacks.com/index.php/shop/arduino/starter-kits/arduino-starter-kit.html

1.2.4 Arduino Sidekick Basic kit

Store website: http://www.seeedstudio.com/depot/arduino-sidekick-basic-kit-p-775.html

Alternative online store

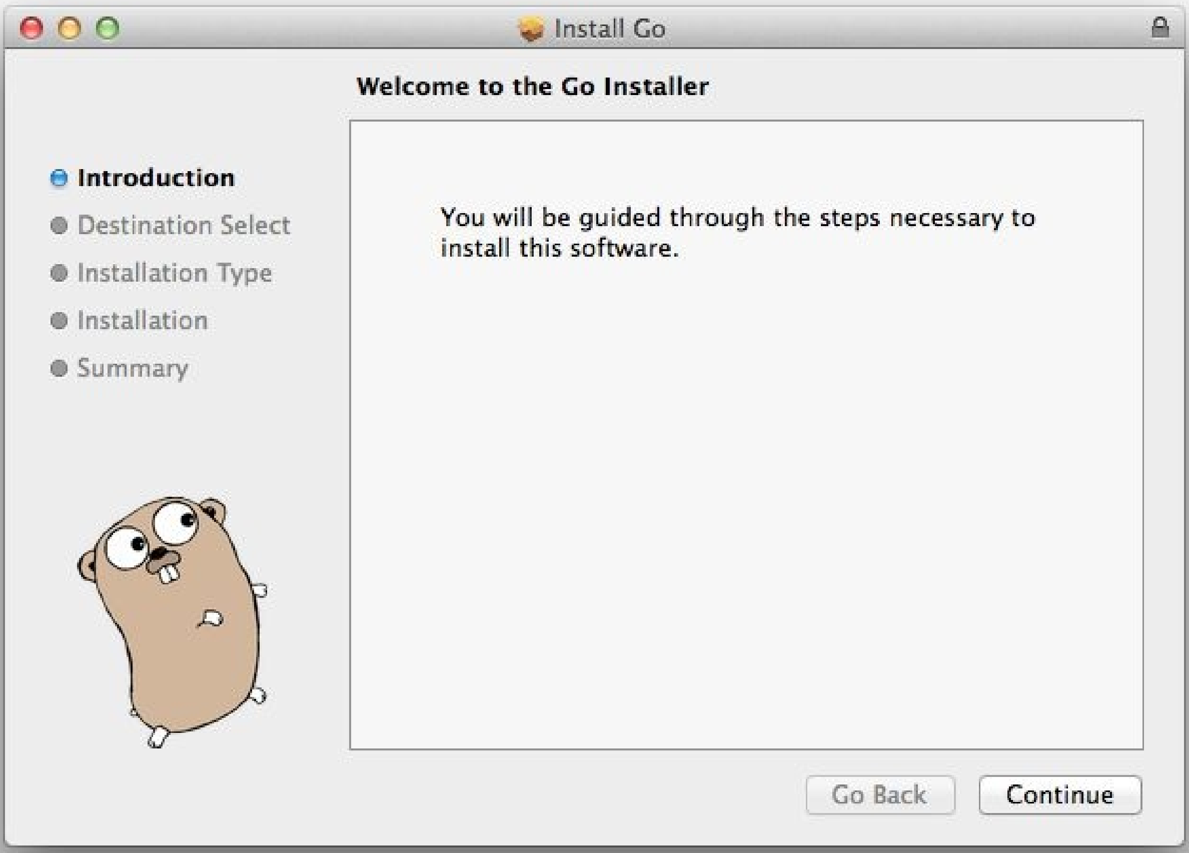
http://www.amazon.com/Arduino-Sidekick-Basic-Kit-Version/dp/B007B14HM8/

http://www.exp-tech.de/Zubehoer/Arduino-Sidekick-Basic-Kit.html

1.3 Go

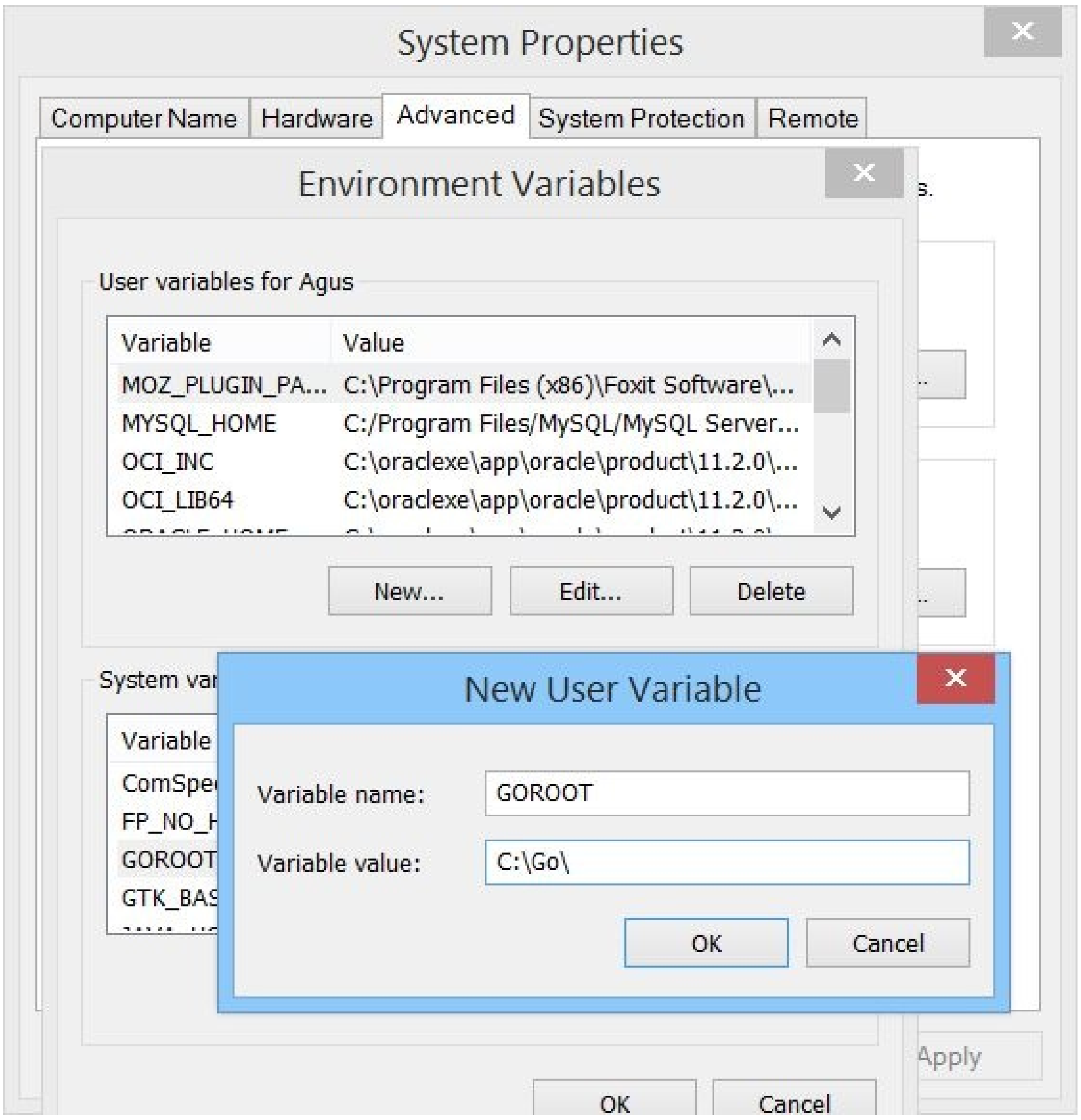
The official web of Go could be found on https://golang.org/. What is Go? Based on information from website, we could know what it is. Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.

Installation of Go application is easy. For Windows and Mac Platform, you download setup file from Go website, http://golang.org/doc/install. Run it and follow installation commands.



You will be guided through the steps necessary to install this software.

The next step is to configure GOROOT path. For Windows platform, you can add GOROOT variable on **Environment Variables**. For Mac/Linux, you can it on your bash profile.

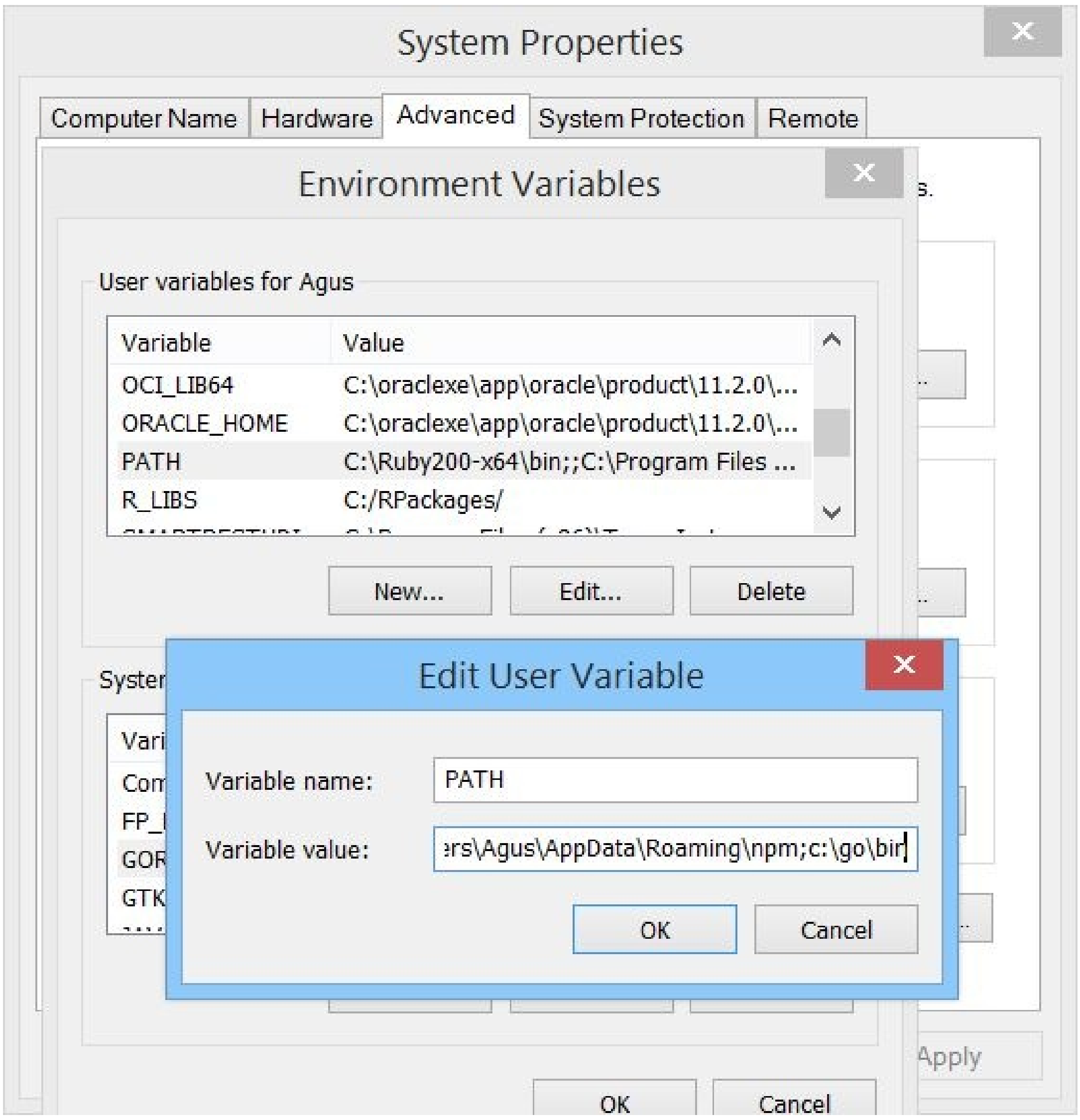


Variable	Value
MOZ_PLUGIN_PA...	C:\Program Files (x86)\Foxit Software\...
MYSQL_HOME	C:/Program Files/MySQL/MySQL Server...
OCI_INC	C:\oraclexe\app\oracle\product\11.2.0\...
OCI_LIB64	C:\oraclexe\app\oracle\product\11.2.0\...

Variable name: GOROOT

Variable value: C:\Go\

For Windows platform, you can add GO installation path, for instance my Go installation path is c:/go/bin, into PATH variable on **Environment Variables**.



User variables for Agus

Variable	Value
OCI_LIB64	C:\oraclexe\app\oracle\product\11.2.0\...
ORACLE_HOME	C:\oraclexe\app\oracle\product\11.2.0\...
PATH	C:\Ruby200-x64\bin;;C:\Program Files ...
R_LIBS	C:/RPackages/

Edit User Variable

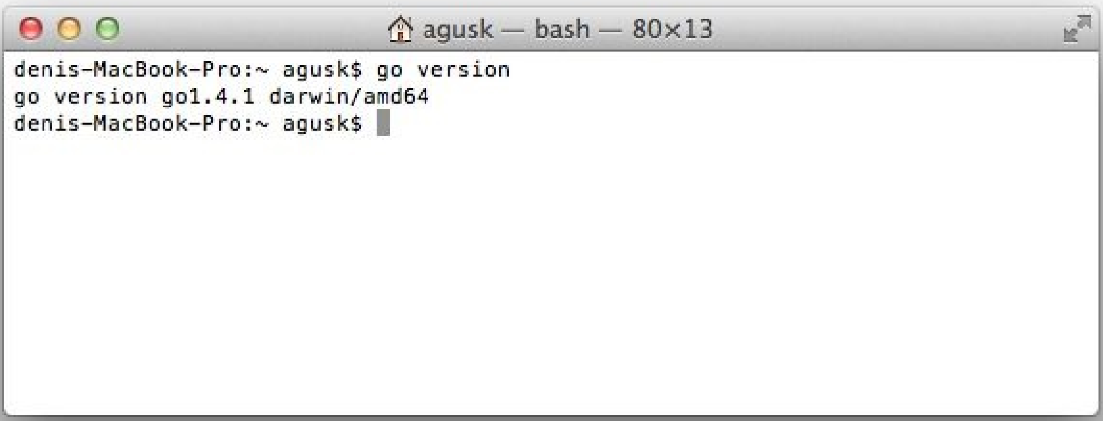
Variable name: PATH

Variable value: ɜrs\Agus\AppData\Roaming\npm;c:\go\bin

After configured, you can verify Go version by typing this command on your Terminal or CMD for Windows.

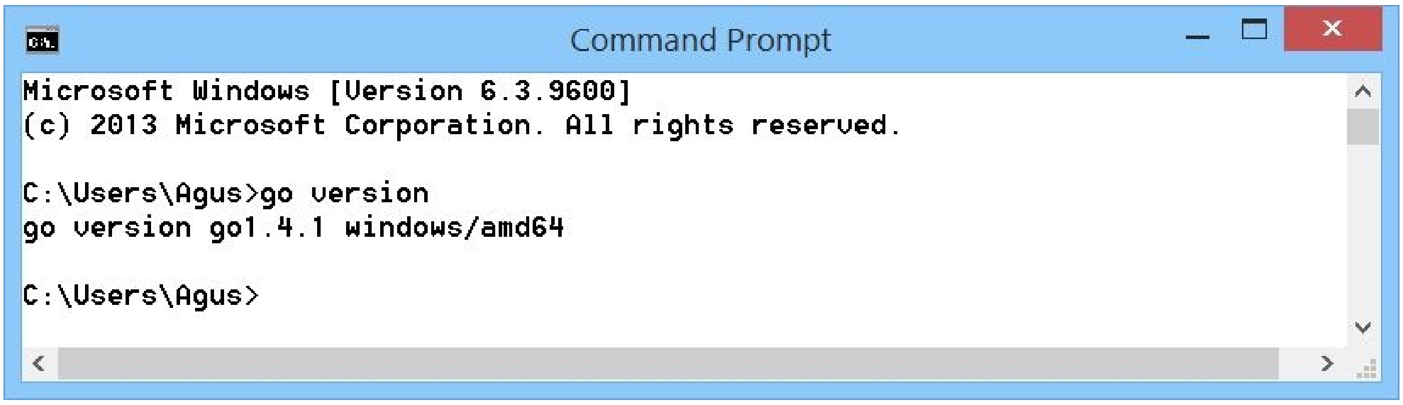
```
$ go version
```

A sample output can be seen in Figure below, Mac platform.



```
denis-MacBook-Pro:~ agusk$ go version
go version go1.4.1 darwin/amd64
denis-MacBook-Pro:~ agusk$
```

The output of program on Windows platform.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Agus>go version
go version go1.4.1 windows/amd64

C:\Users\Agus>
```

In this book, I don't explore all programming language for Go. You can read it on several book or website. I have already written a book about Go Programming by Example. You can get this book on this link, http://blog.aguskurniawan.net/post/Go-Programming-by-Example.aspx .

1.4 Arduino Software

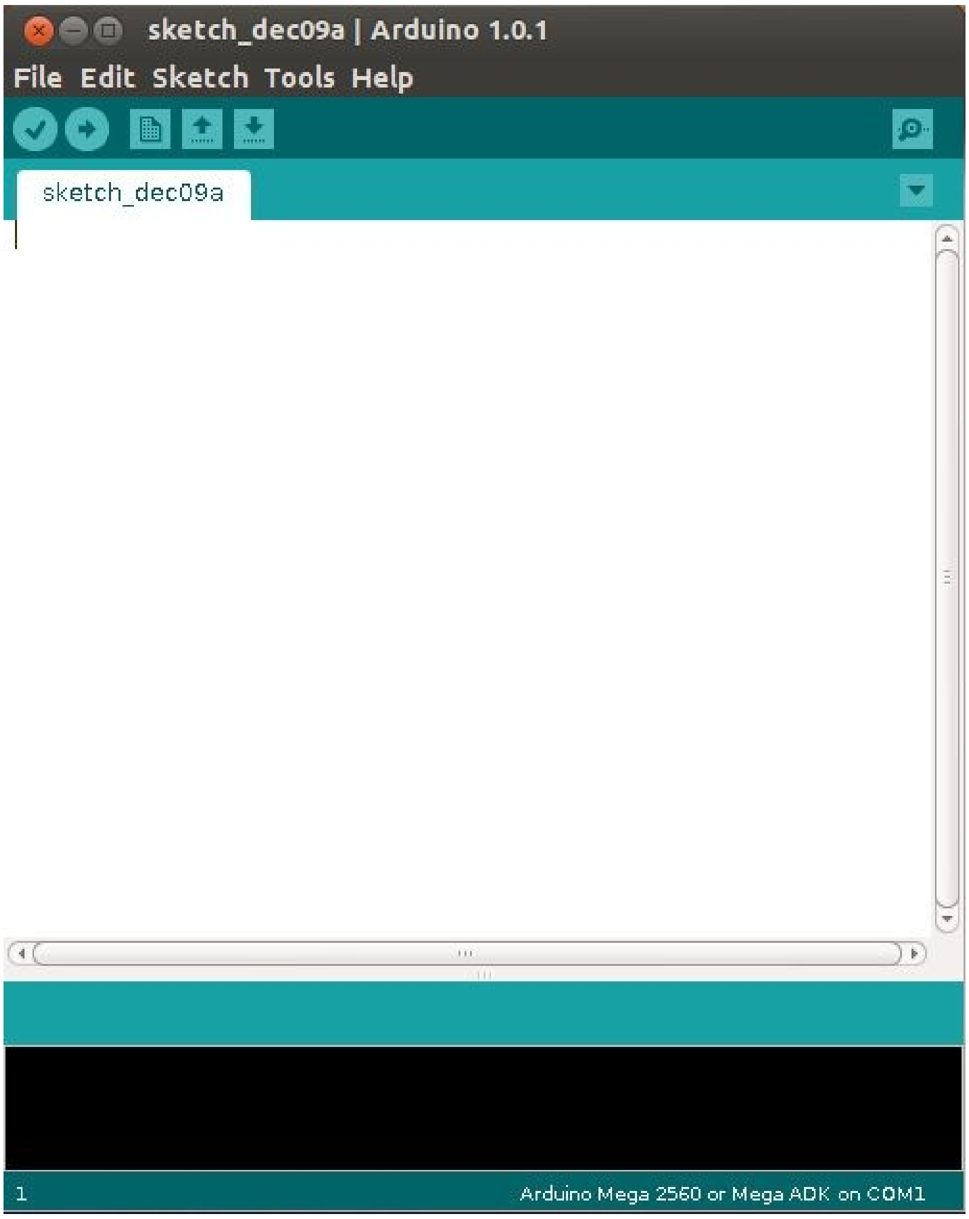
To develop application based on Arduino board, we need Arduino software. You can obtain it on http://arduino.cc/en/Main/Software . Please install based on your platform.

If your platform is Ubuntu, you can install by writing this script

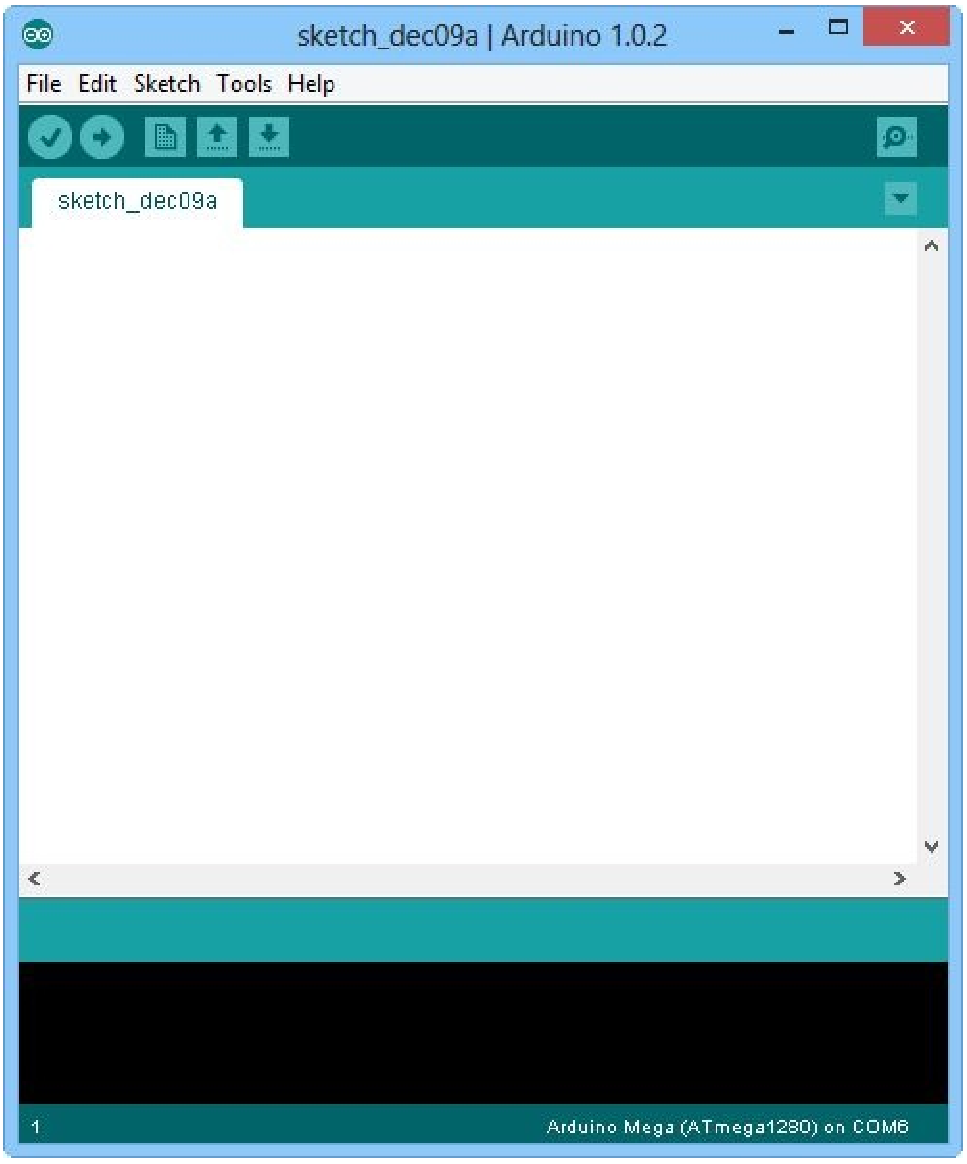
```
sudo apt-get install arduino
```

For Windows platform, you can download setup file and install it.

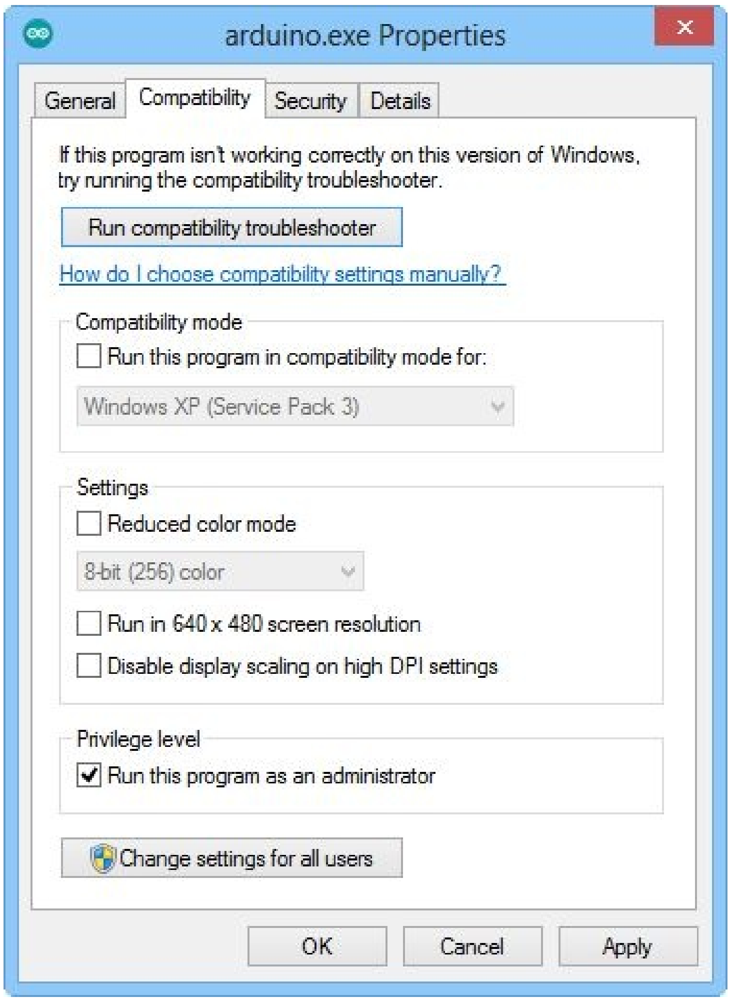
The following is a screenshot of Arduino software on Ubuntu platform.


sketch_dec09a | Arduino 1.0.1
File Edit Sketch Tools Help
sketch_dec09a
1
Arduino Mega 2560 or Mega ADK on COM1

Here is Arduino software on Windows 8 platform.


sketch_dec09a | Arduino 1.0.2
File Edit Sketch Tools Help
sketch_dec09a
1
Arduino Mega (ATmega1280) on COM6

If you run Arduino software on Windows platform, you should configure **arduino.exe** running as Administrator. You can change it by editing file property. Click **Compatibility** and then checked **Run this program as an administrator**.


arduino.exe Properties

General | Compatibility | Security | Details

If this program isn't working correctly on this version of Windows, try running the compatibility troubleshooter.

Run compatibility troubleshooter

How do I choose compatibility settings manually?

Compatibility mode

☐ Run this program in compatibility mode for:

Windows XP (Service Pack 3)

Settings

☐ Reduced color mode

8-bit (256) color

☐ Run in 640 x 480 screen resolution

☐ Disable display scaling on high DPI settings

Privilege level

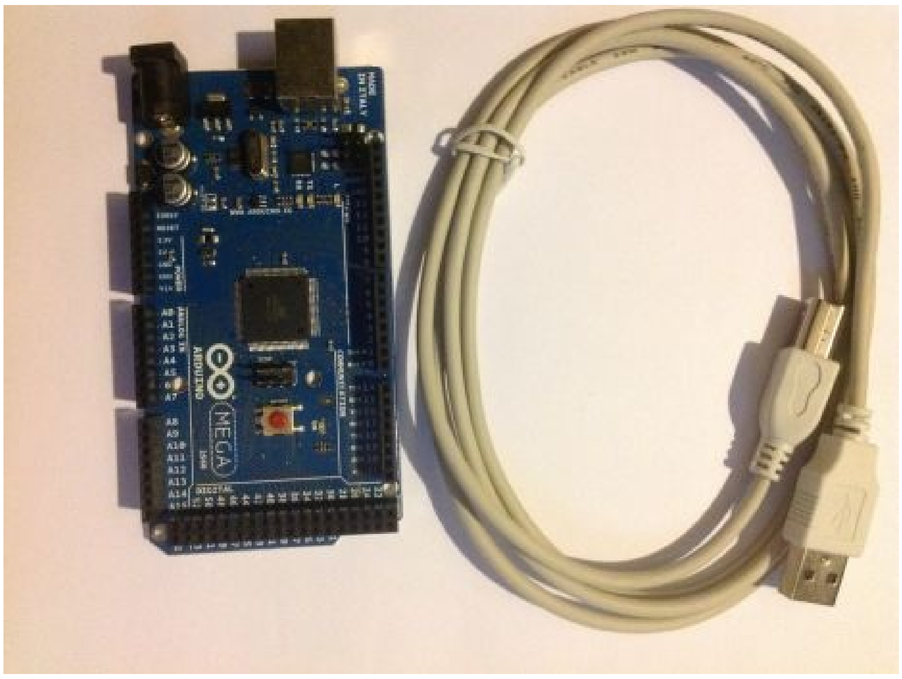
☑ Run this program as an administrator

Change settings for all users

OK | Cancel | Apply

1.5 Testing

For testing, I used Arduino Uno R3 and Arduino Mega 2560 on Ubuntu and Windows 8.1 platforms.



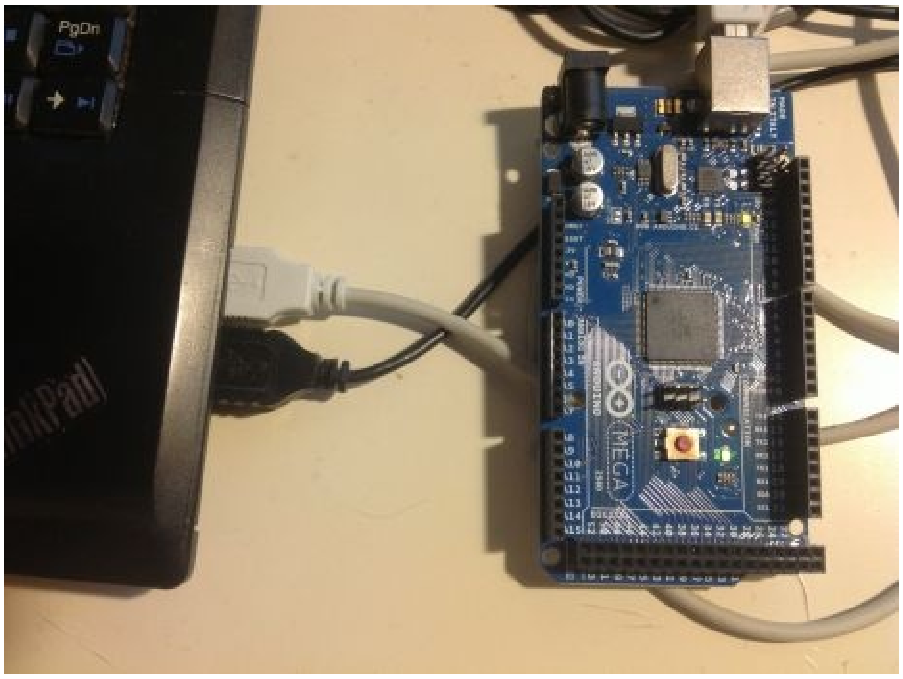
I also used Arduino Sidekick Basic kit for electronic components.

2. Hello World: Arduino and Go

This chapter explains how to work with Arduino and Go for getting started.

2.1 Arduino World

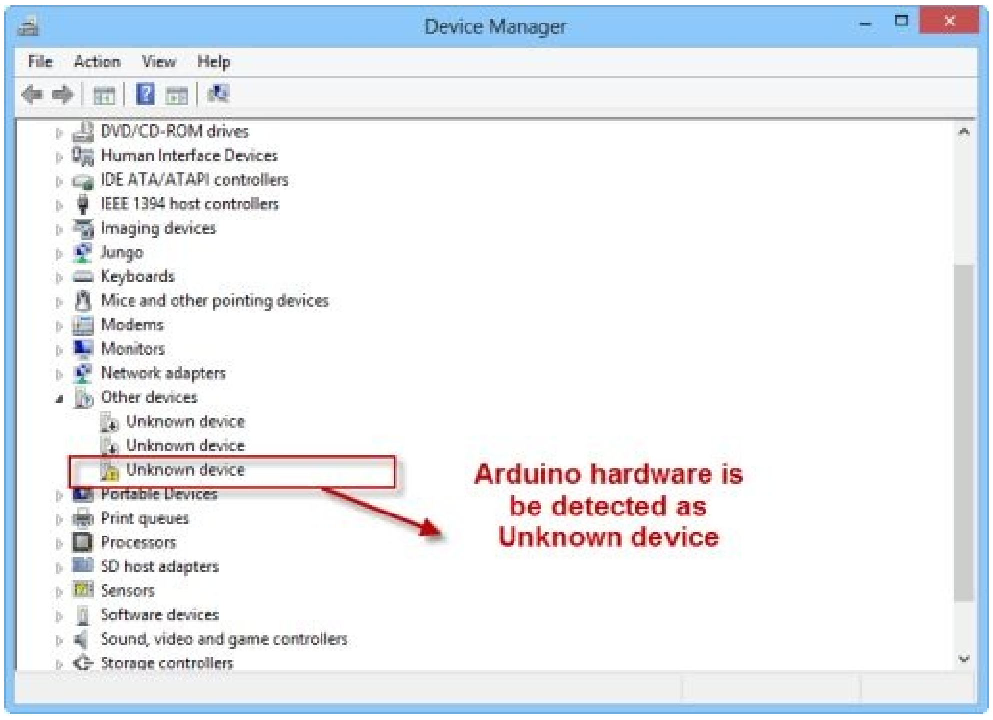
After you installed Arduino software, you can plugin Arduino hardware into computer via USB.



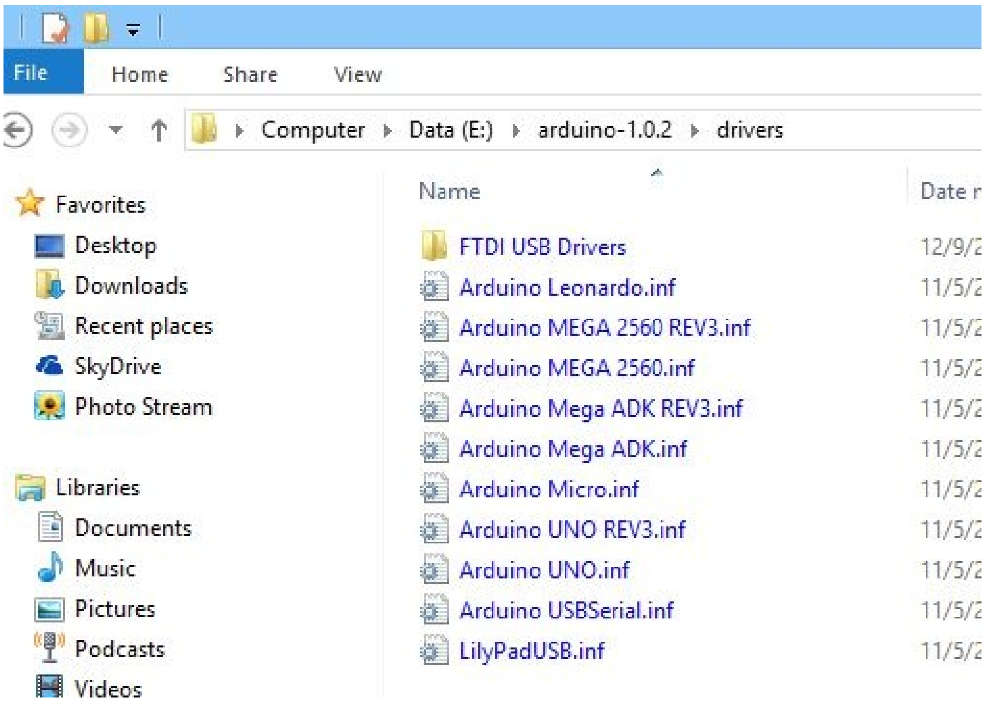
Then you execute Arduino software. In general it will detect Arduino hardware include Arduino type and model.

2.1.1 Arduino Hardware Driver on Windows 8/8.1

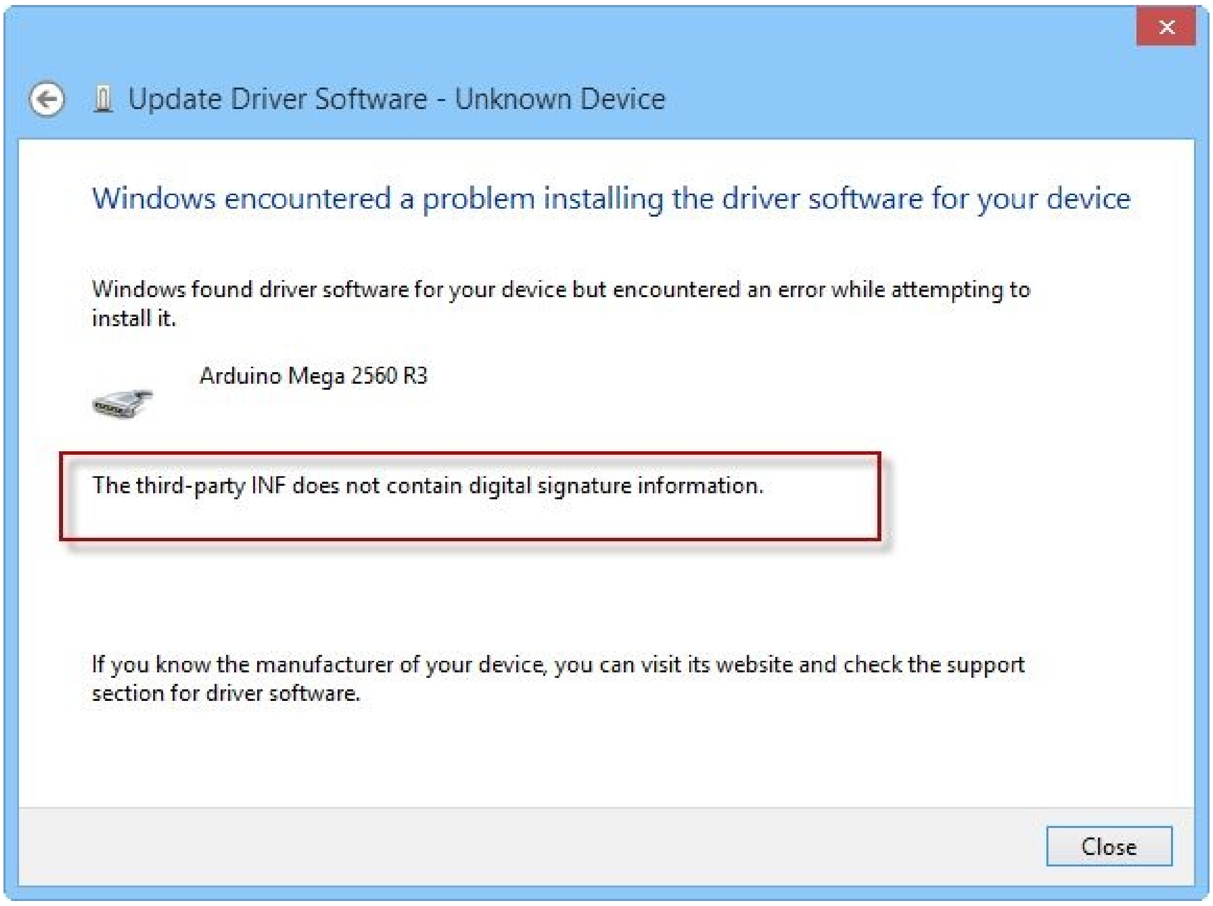
On Windows platform, you may get a problem about Arduino hardware driver. It doesn’t be recognized on Windows Device Manager as below.



You can update this device driver by navigating hardware driver on the **driver** folder of Arduino software installation folder, for instance **E:\arduino-1.0.2\drivers** .



Sometime you may get a problem especially for x64 edition about digital signature of Arduino driver, shown in Figure below.



Windows encountered a problem installing the driver software for your device

Windows found driver software for your device but encountered an error while attempting to install it.

Arduino Mega 2560 R3

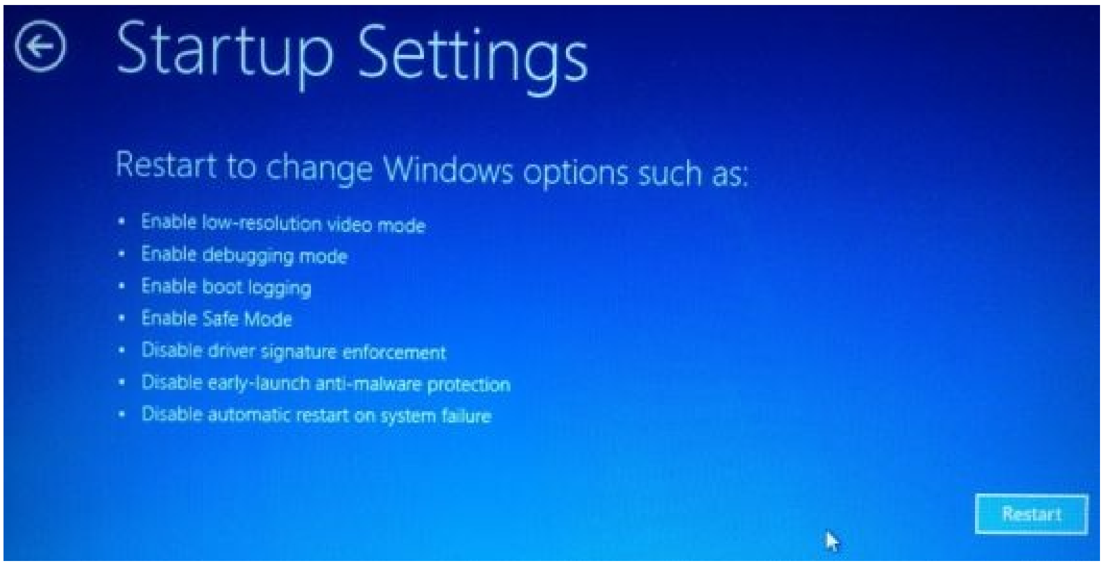
The third-party INF does not contain digital signature information.

If you know the manufacturer of your device, you can visit its website and check the support section for driver software.

Configure to ignore the digital signature settings on your Windows. Normally you can do it by pressing F8 key on restarting Windows. Then choose **Disable Driver signature Enforcement**.

For Windows 8, you can do by clicking Restart menu and pressing SHIFT key. Select Troubleshoot. Then select Advanced options.

On Advanced options display, select Startup settings, shown in Figure below.



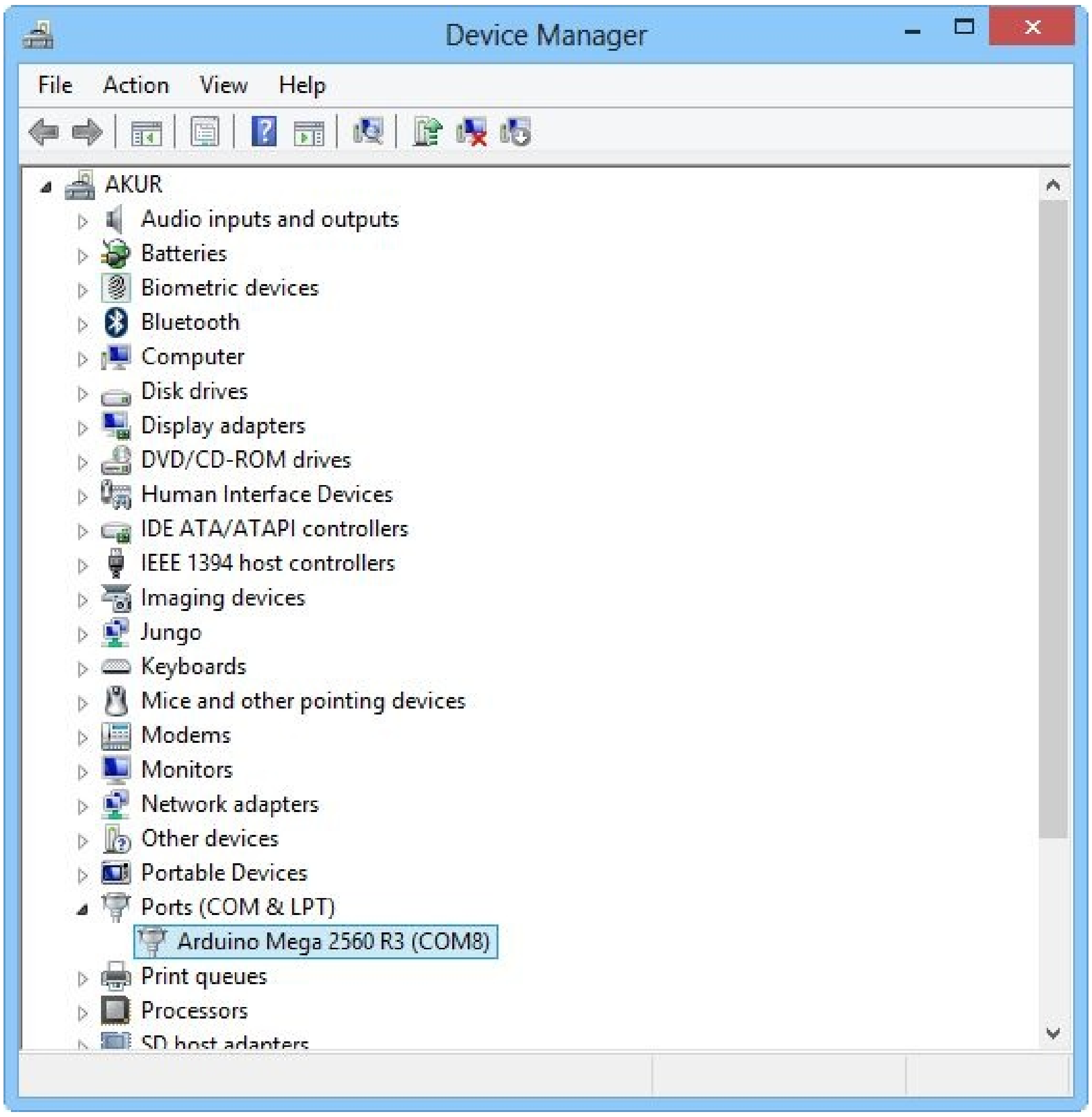
Restart to change Windows options such as:

- Enable low-resolution video mode
- Enable debugging mode
- Enable boot logging
- Enable Safe Mode
- Disable driver signature enforcement
- Disable early-launch anti-malware protection
- Disable automatic restart on system failure

Click **Restart** button. Then Windows 8 will restart. After that, select **Disable driver signature enforcement**.

Now you can install Arduino driver on your Windows.

If success, you can see Arduino hardware on The **Device Manager** as below.

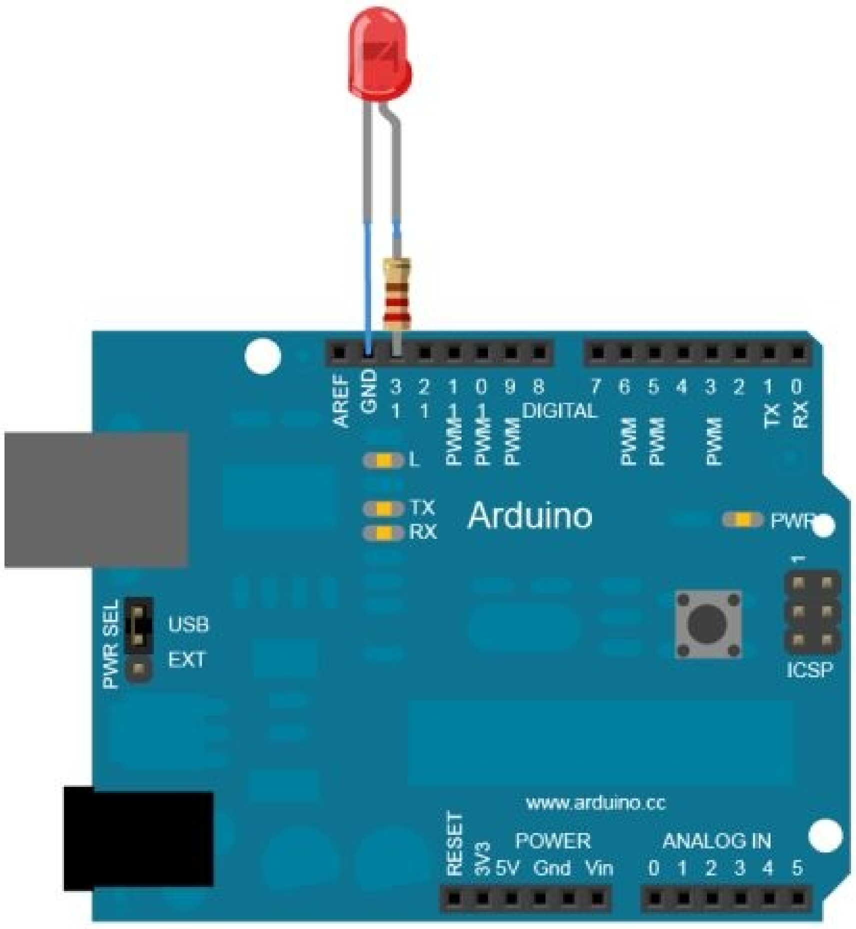


You can see your Arduino hardware running on COM8.

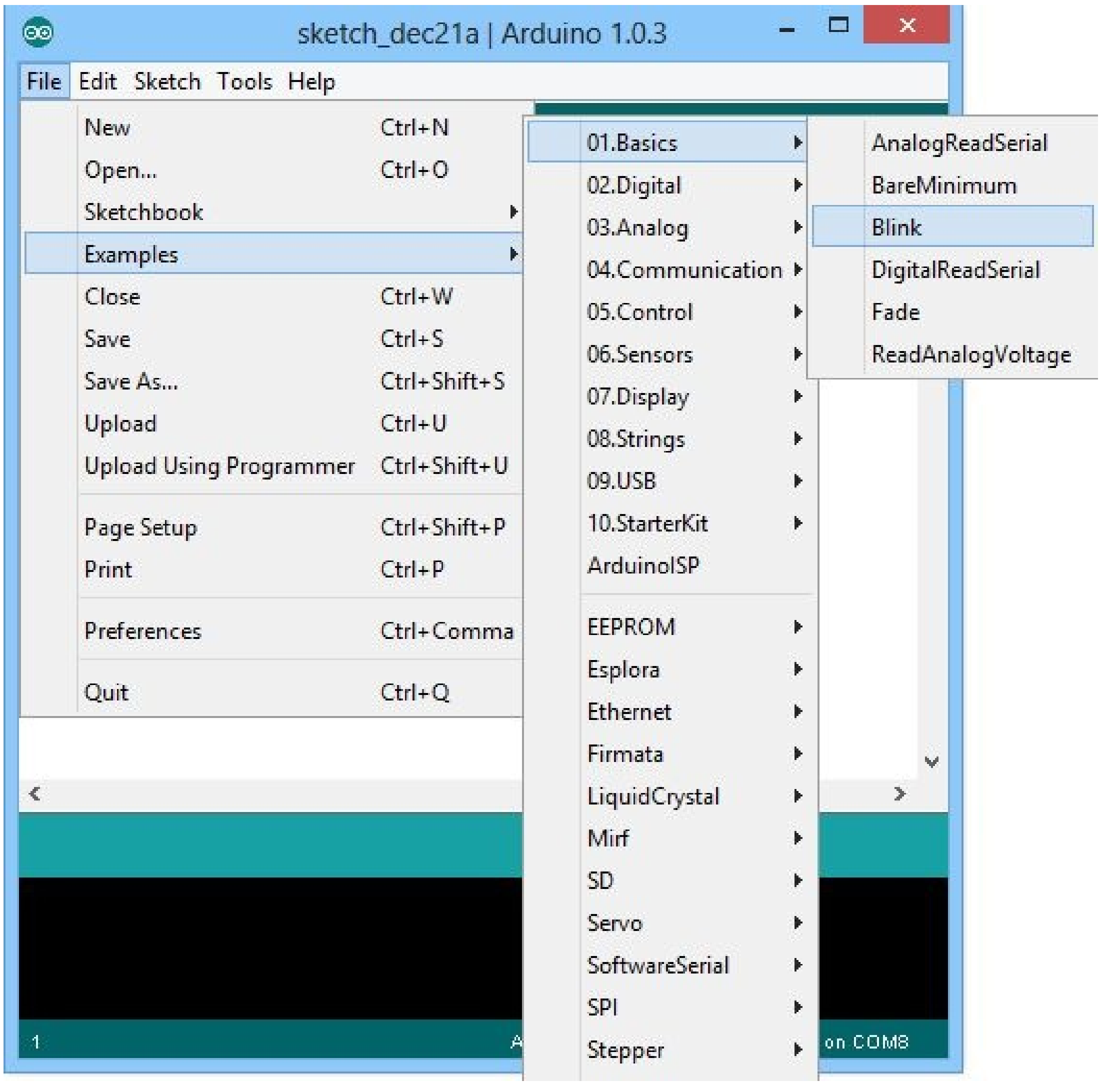
2.1.2 Simple Testing

Now you're ready to get started. For illustration, I use the sample code from Arduino, Blink. You can visit on http://arduino.cc/en/Tutorial/Blink for configuration.

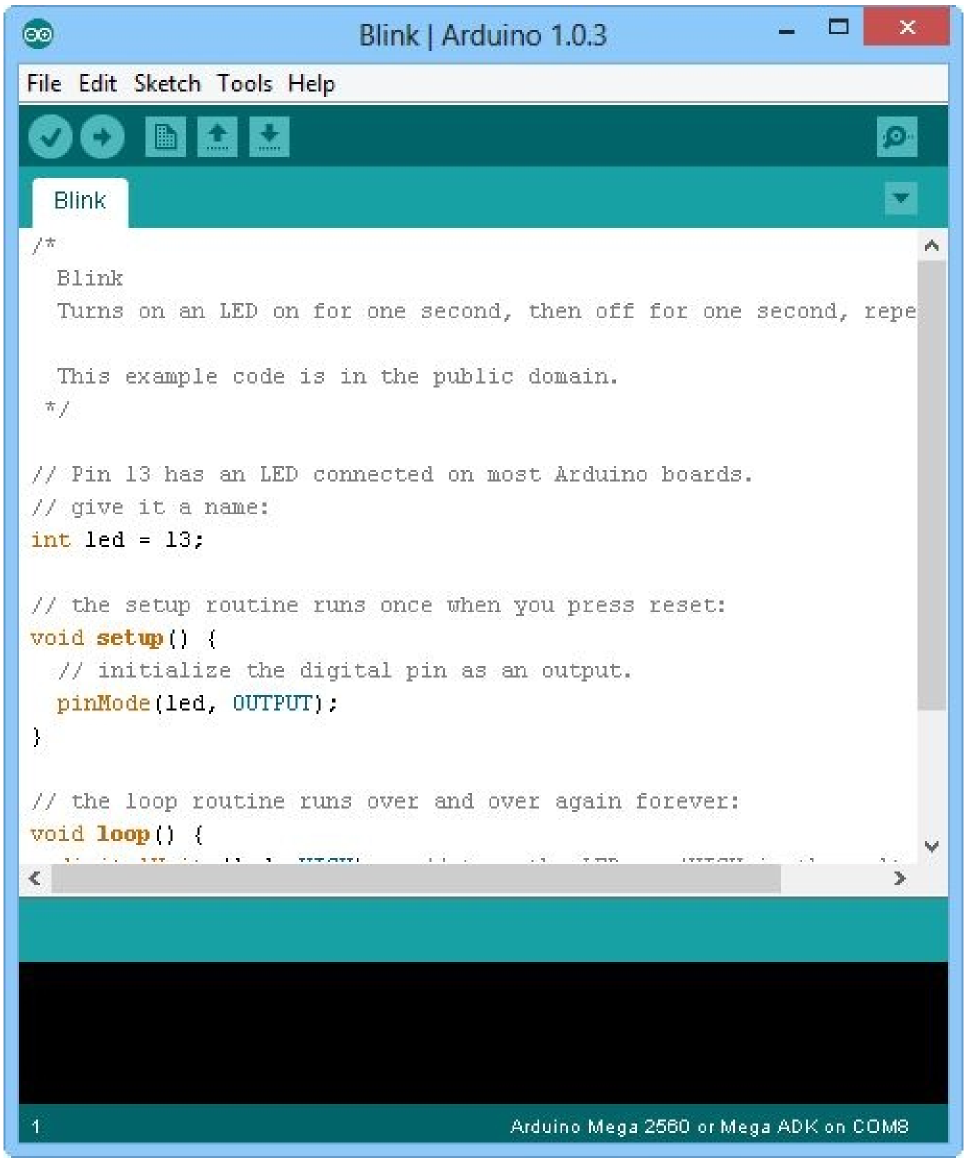
For this scenario, you need LED and resistor. Attach resistor to pin 13. Negative pin of LED is attached to pin GND. You can see the hardware configuration as below.



On Arduino software, Click **File** -> **Examples** -> **01.Basics** -> **Blink**.

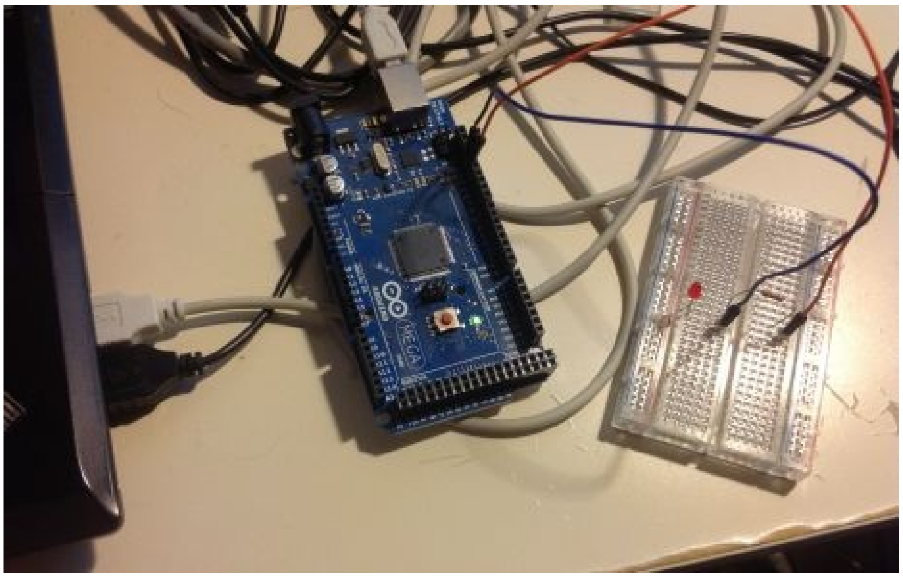


Then you will get a sample code of Blink app.


Blink | Arduino 1.0.3
File Edit Sketch Tools Help
Blink
1
Arduino Mega 2560 or Mega ADK on COM8

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
```

Now try to connect your Arduino into computer.



Compile and upload Blink app to Arduino hardware. If success, you can see the LED will be on/off every second.

2.2 Arduino and Go

To communicate between Arduino and computer, we can use a serial communication. It means Go gives command and receives data via serial port.

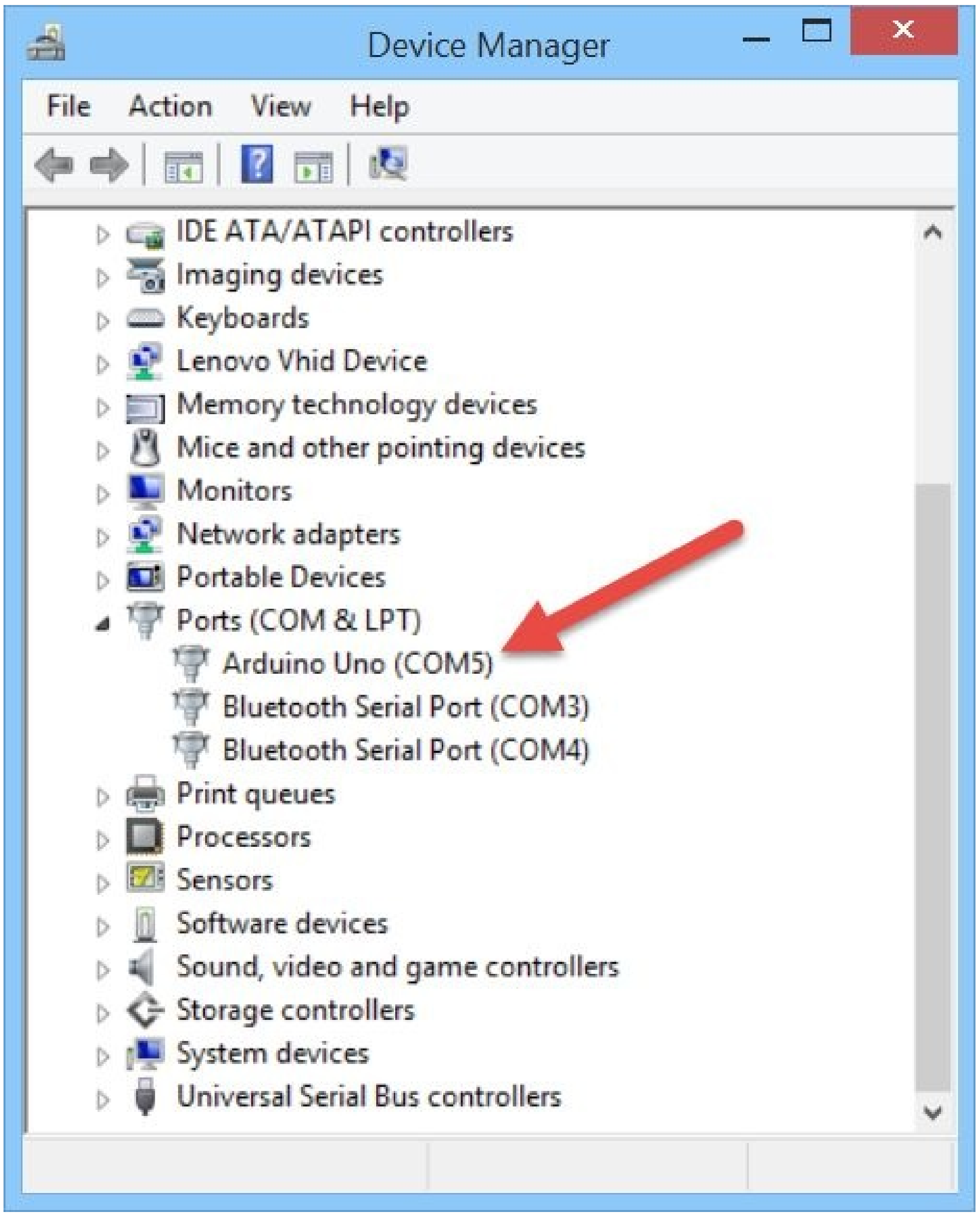
On Go, we can use several Go packages. The following is a list of Go package for communicating with serial port:

- GoSerial, https://code.google.com/p/goserial/
- go-serial, https://github.com/jacobsa/go-serial

In this section, I just show you how to use GoSerial to communicate with serial port.

2.3 Testing Serial Port using Go

Now you can connect your Arduino to computer. For Windows platform, you can verify by opening Device Manager and expand Ports (COM & LPT). You can see COM your Arduino used.



For Debian/Ubuntu/Mac, you can check it by typing on terminal

```
ls /dev/ttyACM*
```

You can see Arduino serial port. Here is a sample output.

```
/dev/ttyACM0
```

Now we create project called serialtest by creating a folder, serialtest.

```
$ mkdir serialtest
$ cd serialtest
```

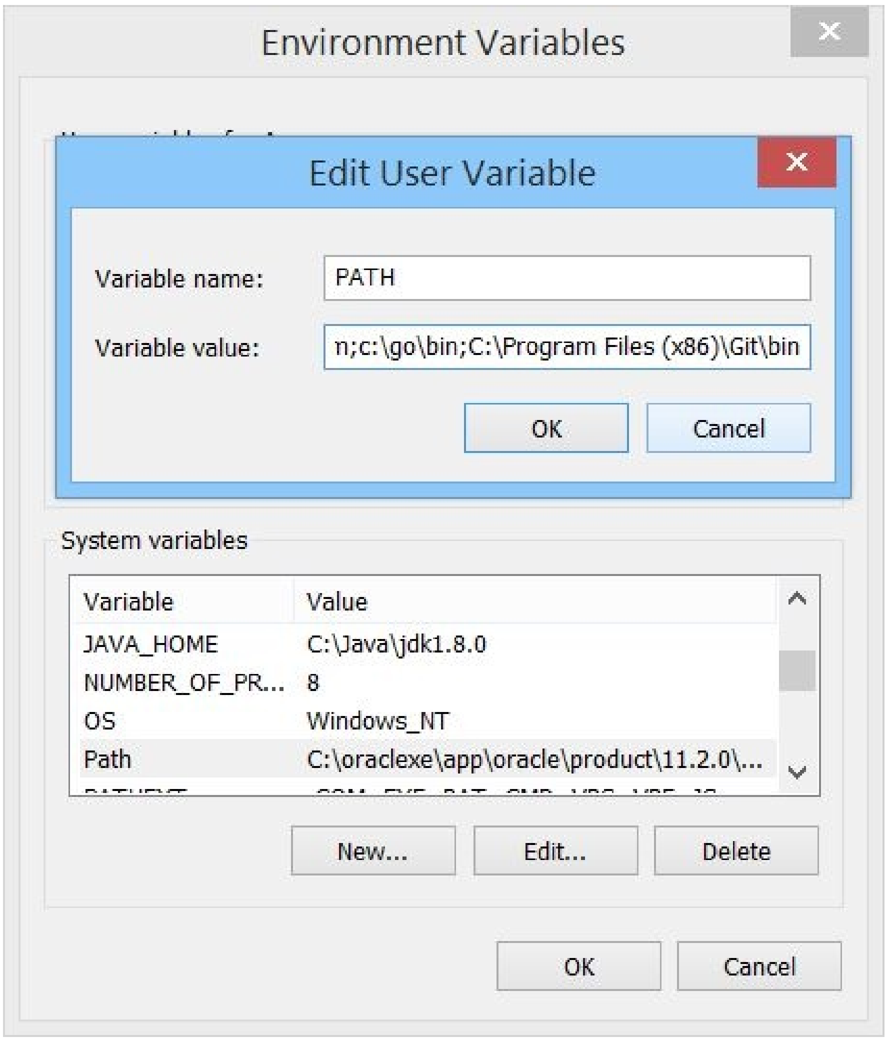
Create a file, called **main.go**. Write the following code.

```
package main

import (
    "github.com/tarm/goserial"
    "fmt"
)

func main() {
    c := &serial.Config{Name: "COM6", Baud: 9600}
    s, err := serial.OpenPort(c)
    if err != nil {
        panic(err)
    }
    defer s.Close()
    fmt.Println("connected")
}
```

Note: change your serial port. In this section, my Arduino is connected to COM6 in Windows platform.

Because we use external package from Github, you must install git runtime.

To install Go library from Github, you must instal git, http://git-scm.com/ . Download it based on your platform. If you use Windows platform, don’t forget to set it on PATH on Environment Variables.



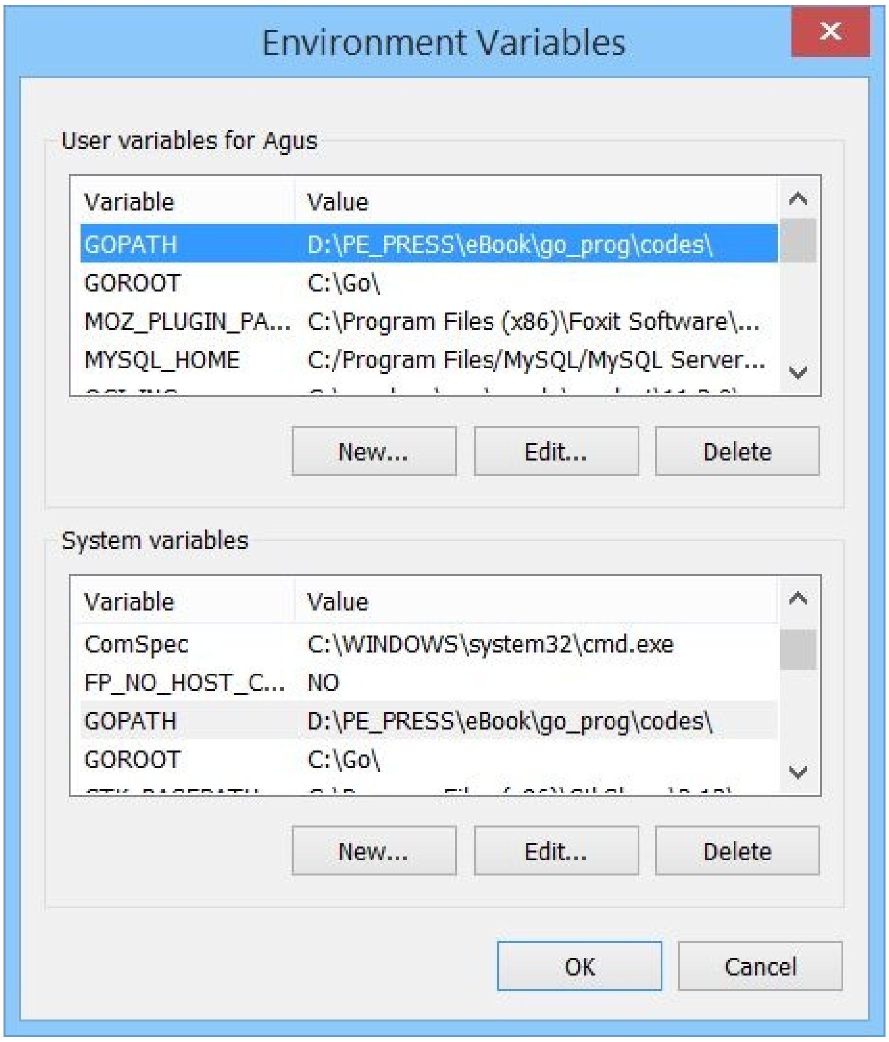
Variable	Value
JAVA_HOME	C:\Java\jdk1.8.0
NUMBER_OF_PR...	8
OS	Windows_NT
Path	C:\oraclexe\app\oracle\product\11.2.0\...

The next step is to configure GOPATH. It represents our workspace path, for instance, my workspace path is **D:\PE_PRESS\eBook\arduino_go\codes**. In Linux/Mac, you can define your own workspace path in under your account home path, for instance. Further information, you can read it on https://golang.org/doc/code.html .

If you use Linux/Mac, you can define GOPATH using export command or you add it on your profile file.

```
$ mkdir $HOME/go
$ export GOPATH=$HOME/go
```

If you use Windows platform, you open **Environment Variables**. You can open it from **Advanced system settings**. Then, click **Environment Variables** button. Add GOPATH on your user and System variables.


Environment Variables

User variables for Agus

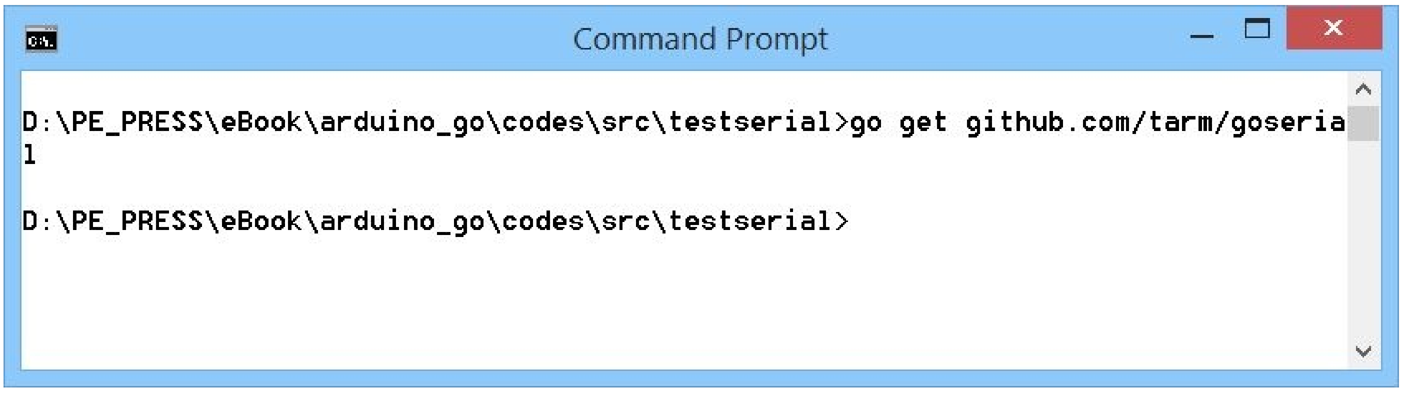
Variable	Value
GOPATH	D:\PE_PRESS\eBook\go_prog\codes\
GOROOT	C:\Go\
MOZ_PLUGIN_PA...	C:\Program Files (x86)\Foxit Software\...
MYSQL_HOME	C:/Program Files/MySQL/MySQL Server...

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...	NO
GOPATH	D:\PE_PRESS\eBook\go_prog\codes\
GOROOT	C:\Go\

Now we can install goserial library from Github. Type the following command.

```
$ go get github.com/tarm/goserial
```



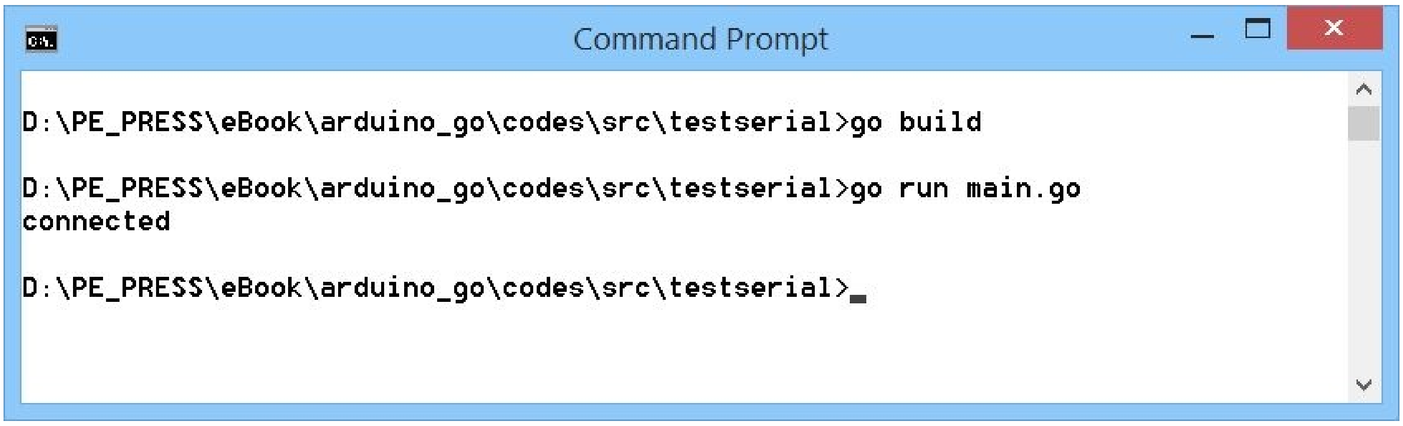
```
D:\PE_PRESS\eBook\arduino_go\codes\src\testserial>go get github.com/tarm/goseria
l

D:\PE_PRESS\eBook\arduino_go\codes\src\testserial>
```

Save this code on **main.go** file.

Now you can build and run this program. Don't forget to connect your Arduino into PC via USB.

```
$ go build
$ go run main.go
```

A program output can be seen in Figure below.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\testserial>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\testserial>go run main.go
connected

D:\PE_PRESS\eBook\arduino_go\codes\src\testserial>_
```

2.4 Testing for Arduino and Go

Now we test for communicating between Arduino and Go. In this scenario, we build Arduino app to send data to serial port. Then, Go app will receive this message by listening on serial port.

The first step is to create Arduino app. From Blink Arduino app, you can modify this code. Write this code:

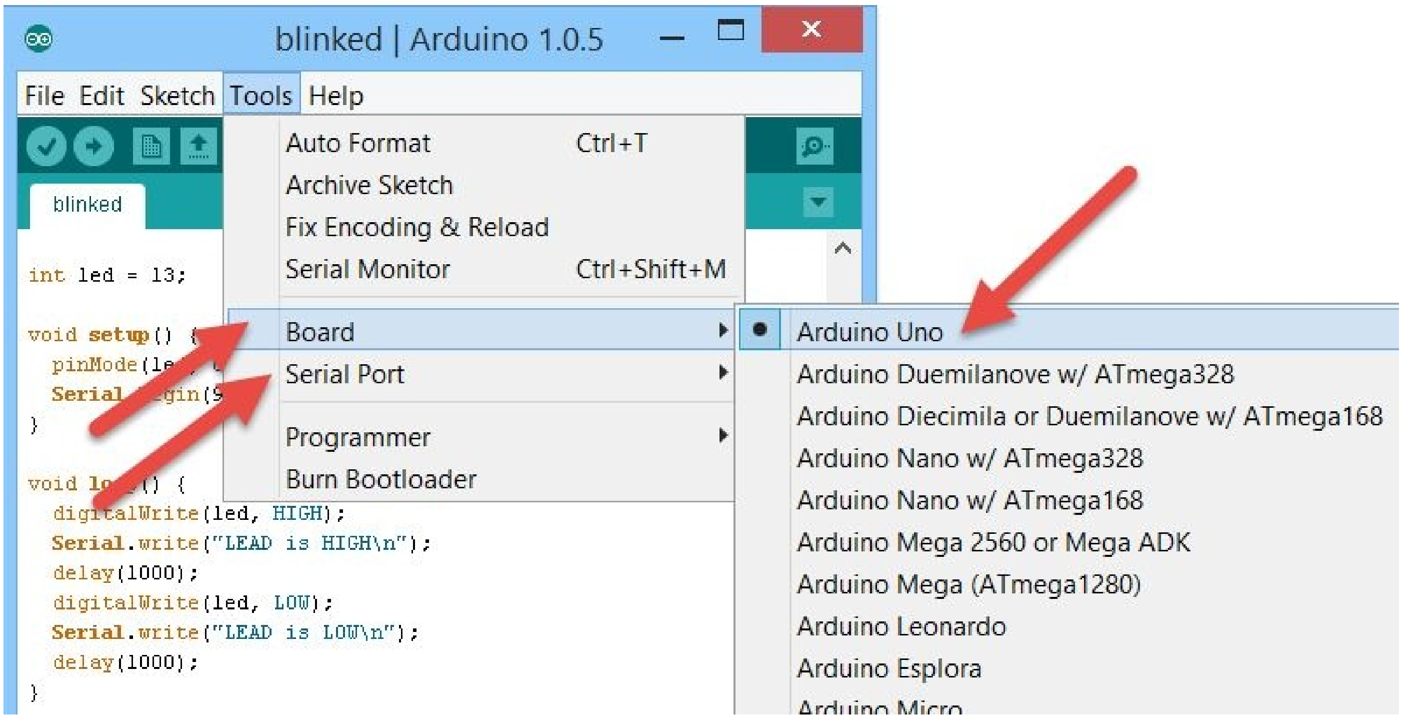
```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(led, HIGH);
  Serial.write("LEAD is HIGH\n");
  delay(1000);
  digitalWrite(led, LOW);
  Serial.write("LEAD is LOW\n");
  delay(1000);
}
```

Explanation

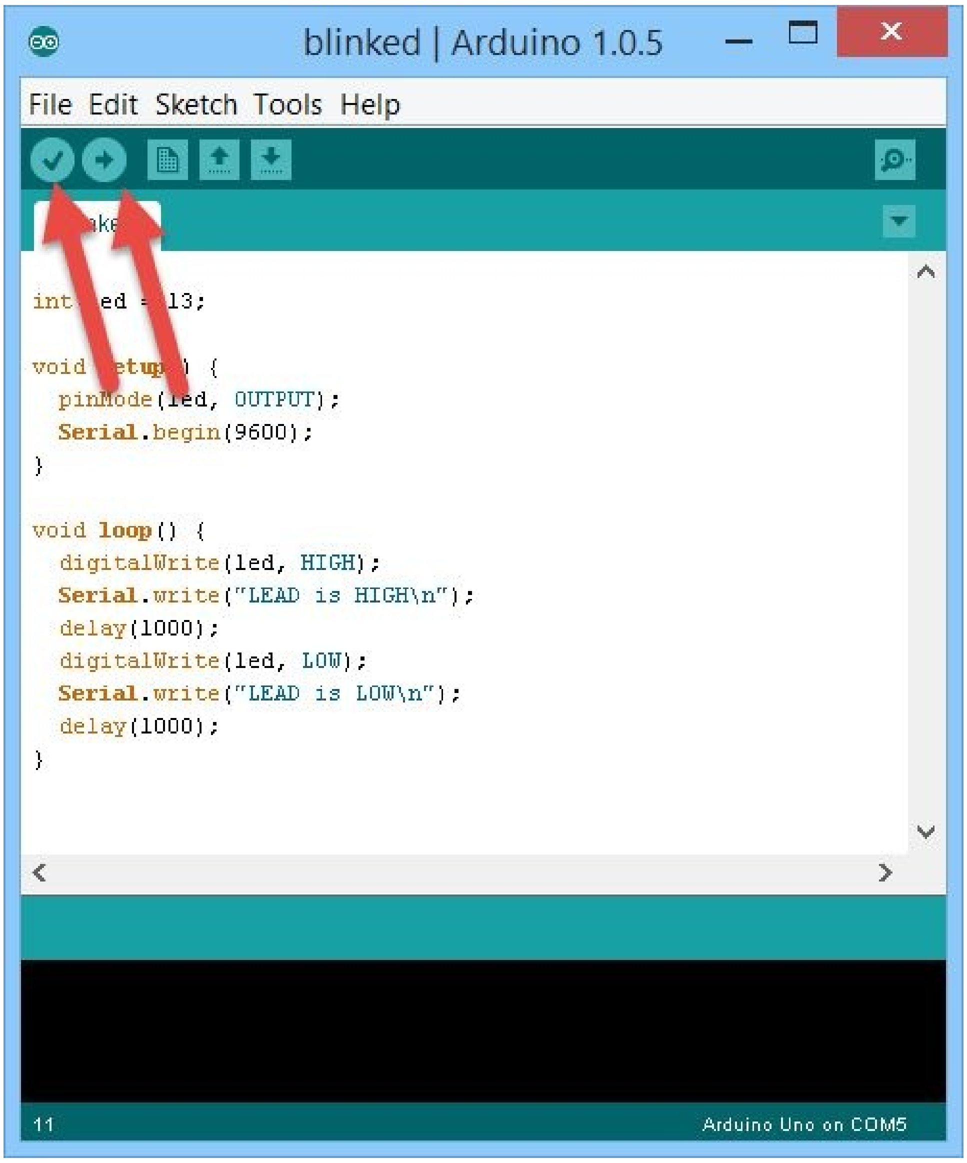
- On **setup()**, we activate serial port on 9600 baud rate and LED on pin 13
- On **loop()**, we write HIGH on pin 13 and then send a message “LEAD is HIGH\n” to serial port
- We also do it again. It writes LOW value and sends a message “LEAD is LOW\n” to serial port

Save this code as blinked. Compile and upload this code to Arduino hardware.

Make sure you already configure Arduino board and serial port of Arduino board.

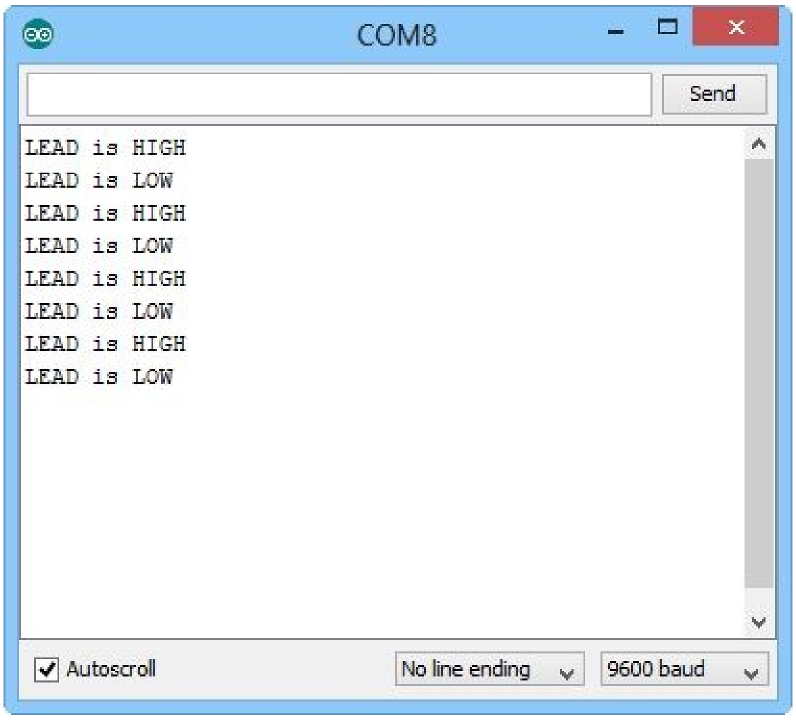

blinked | Arduino 1.0.5
File Edit Sketch Tools Help
blinked
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro

You can compile by clicking icon checked and deploy to Arduino by clicking icon arrow.



```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(led, HIGH);
  Serial.write("LEAD is HIGH\n");
  delay(1000);
  digitalWrite(led, LOW);
  Serial.write("LEAD is LOW\n");
  delay(1000);
}
```

You can check the serial port response using Serial Monitor. Click menu **Tools** -> **Serial Monitor**. You will see messages from Arduino hardware.



```
LEAD is HIGH
LEAD is LOW
LEAD is HIGH
LEAD is LOW
LEAD is HIGH
LEAD is LOW
LEAD is HIGH
LEAD is LOW
```

On the next step, we create Go app. Basically we create a Go app to listen incoming messages from serial port. Create a project, called serialdemo, by creating folder, serialdemo.

```
$ mkdir serialdemo
$ cd serialdemo
```

Create a file, called **main.go**. Write the following code.

```
package main

import (
    "github.com/tarm/goserial"
    "fmt"
    "strings"
)

func main() {
    // open serial port Arduino
    // change port value based on your platform
    c := &serial.Config{Name: "COM6", Baud: 9600}
    s, err := serial.OpenPort(c)
    if err != nil {
        panic(err)
    }
    defer s.Close()
    // read data in background
    go func() {
        buf := make([]byte, 128)
        str := ""
        for  {
            n, err := s.Read(buf)
            if err != nil {
                panic(err)
            }
            str = fmt.Sprintf("%s%s",str,string(buf[:n]))
            if strings.Index(str,"\n")>=0 {
                ind := strings.Index(str,"\n")
                temp := str[:ind+1]
                str = fmt.Sprintf("%s",str[ind+1:])
                fmt.Printf("%s",temp)
            }

        }
    }()


    // press ENTER to exit
    fmt.Println("press Enter to exit..")
    var input string
    fmt.Scanln(&input)
    fmt.Println("done")

}
```

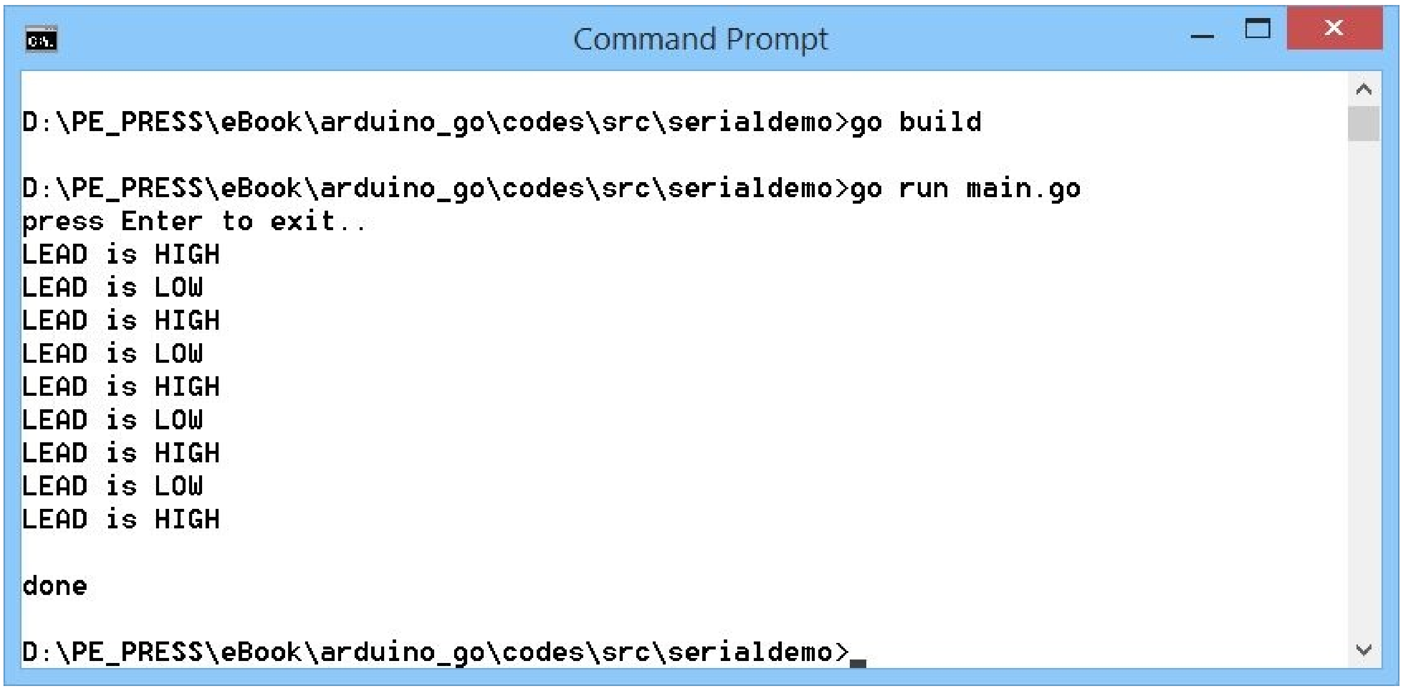
Note: change your Arduino serial port.

Save this code on **main.go** file.

Now you can build and run this program. Don't forget to connect your Arduino into PC via USB.

```
$ go build
$ go run main.go
```

Here is a sample of program output.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\serialdemo>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\serialdemo>go run main.go
press Enter to exit..
LEAD is HIGH
LEAD is LOW
LEAD is HIGH
LEAD is LOW
LEAD is HIGH
LEAD is LOW
LEAD is HIGH
LEAD is LOW
LEAD is HIGH

done

D:\PE_PRESS\eBook\arduino_go\codes\src\serialdemo>_
```

You can see the incoming message from serial port of Arduino. To exit from program, you press ENTER from your PC keyboard.

3. Exploring Go Packages for Arduino

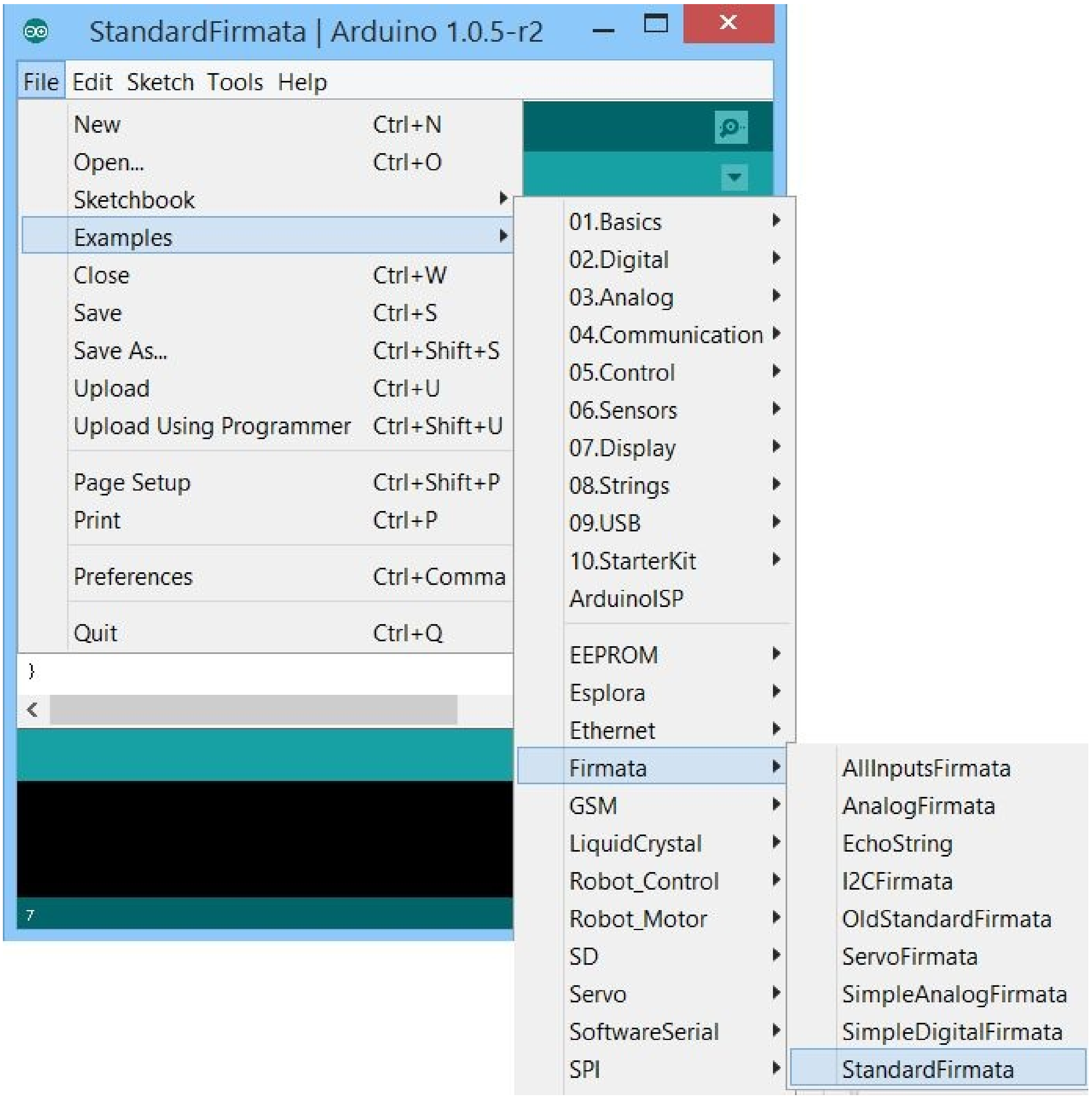
In this chapter I’m going to explain how to use external Go packages to access Arduino directly.

3.1 Getting Started

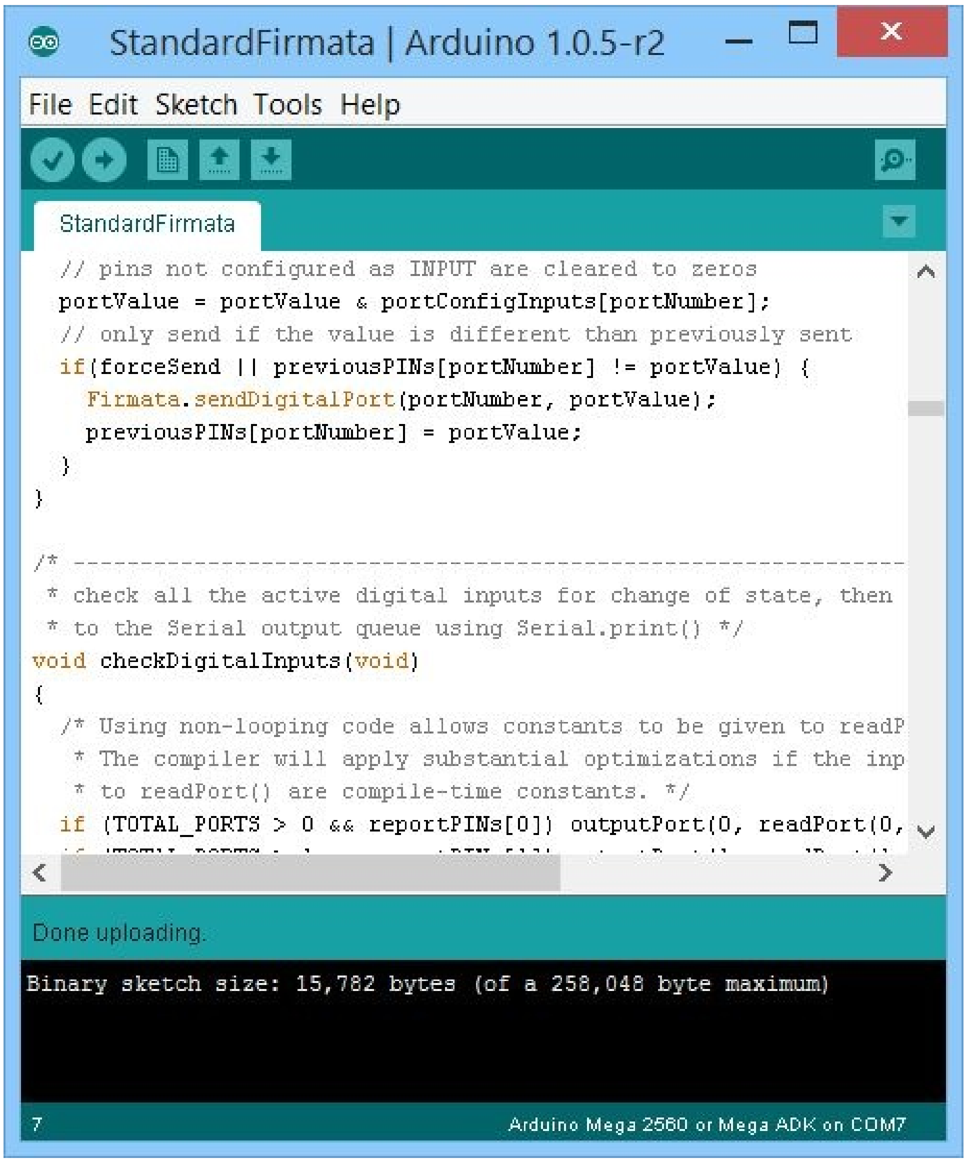
To access Arduino, we can do in many ways. One of them is to use Firmata protocol, http://firmata.org/ . You find many Go packages which implement Firmata protocol. The following is a list of Go package to implement Firmata protocol.

- Gobot, http://gobot.io/
- go-firmata, https://github.com/kraman/go-firmata
- gofirmata, https://github.com/choffee/gofirmata

To use Arduino with Firmata protocol on Arduino, we must load Firmata protocol module on Arduino. Firstly we must load Firmata protocol on your Arduino board. Open Arduino software and click menu File -> Examples -> Firmata ->StandardFirmata .



Then you obtain Firmata codes.


StandardFirmata | Arduino 1.0.5-r2
File Edit Sketch Tools Help
StandardFirmata

```
  // pins not configured as INPUT are cleared to zeros
  portValue = portValue & portConfigInputs[portNumber];
  // only send if the value is different than previously sent
  if(forceSend || previousPINs[portNumber] != portValue) {
    Firmata.sendDigitalPort(portNumber, portValue);
    previousPINs[portNumber] = portValue;
  }
}

/* -----------------------------------------------------------------
 * check all the active digital inputs for change of state, then
 * to the Serial output queue using Serial.print() */
void checkDigitalInputs(void)
{
  /* Using non-looping code allows constants to be given to readP
   * The compiler will apply substantial optimizations if the inp
   * to readPort() are compile-time constants. */
  if (TOTAL_PORTS > 0 && reportPINs[0]) outputPort(0, readPort(0,
```

```
Done uploading.
Binary sketch size: 15,782 bytes (of a 258,048 byte maximum)
```

Now you can compile and deploy Firmata Protocol code into your Arduino board. Don't forget to select your Arduino board model and serial port.

The next step is to build a simple app, blinking led, using Go with Gobot and go-firmata packages.

3.2 Gobot

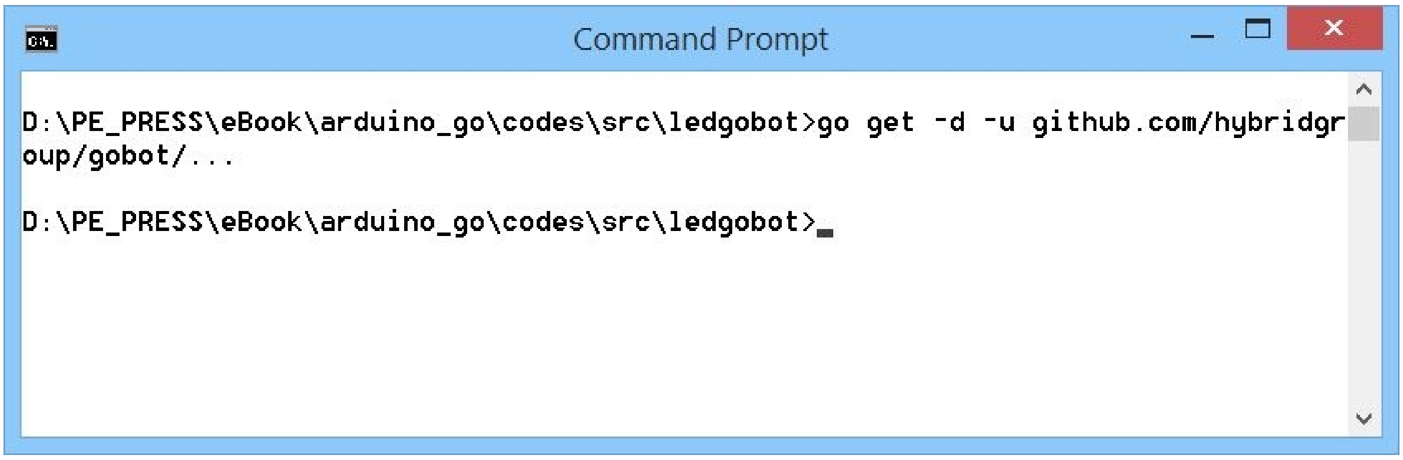
Gobot is a framework for robotics, physical computing, and the Internet of Things, written in the Go programming language. Further information about gobot, you can visit the official website on http://gobot.io/.

In this section, we try to access Arduino via gobot. To install gobot package, you must install the following runtime:

- git, http://git-scm.com/
- mercurial, http://mercurial.selenic.com/wiki/Download

Now you can install gobot package with the following command

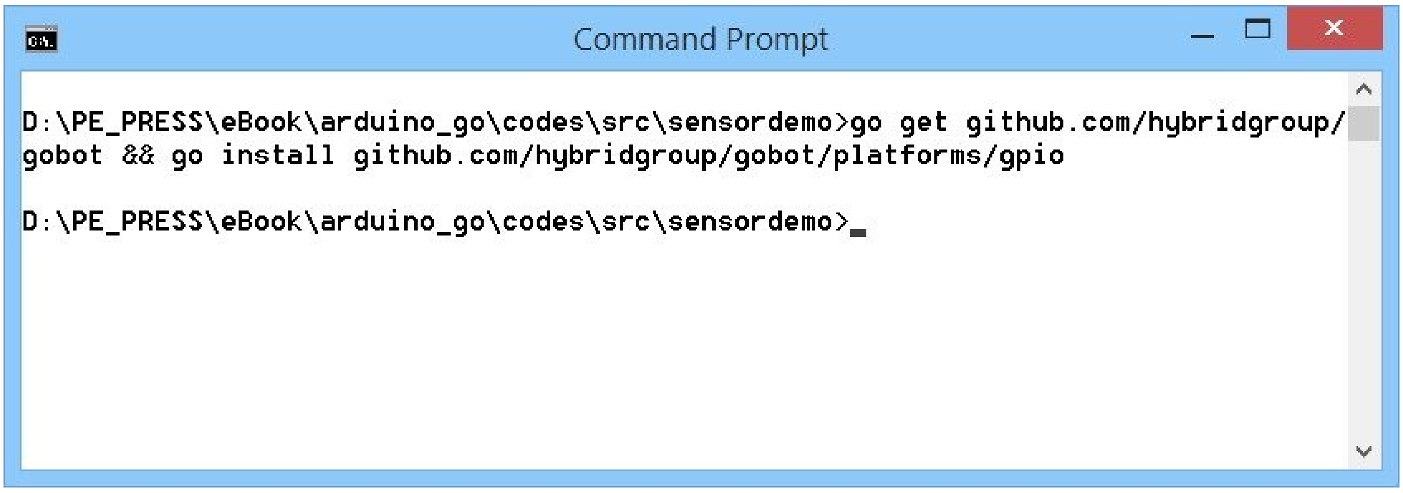
```
$ go get -d -u github.com/hybridgroup/gobot/...
```



```
D:\PE_PRESS\eBook\arduino_go\codes\src\ledgobot>go get -d -u github.com/hybridgr
oup/gobot/...

D:\PE_PRESS\eBook\arduino_go\codes\src\ledgobot>_
```

You also install Firmata for Gobot package. Type this command.

```
$ go get github.com/hybridgroup/gobot && go install github.com/hybridgro
```



```
D:\PE_PRESS\eBook\arduino_go\codes\src\sensordemo>go get github.com/hybridgroup/
gobot && go install github.com/hybridgroup/gobot/platforms/gpio

D:\PE_PRESS\eBook\arduino_go\codes\src\sensordemo>_
```

For illustration, we build a simple app, blinking led. We use LED on the board, LED on pin 13.

Now we create project called ledgobot by creating a folder, ledgobot.

```
$ mkdir ledgobot
$ cd ledgobot
```

Create a file, called **main.go**. Write the following code.

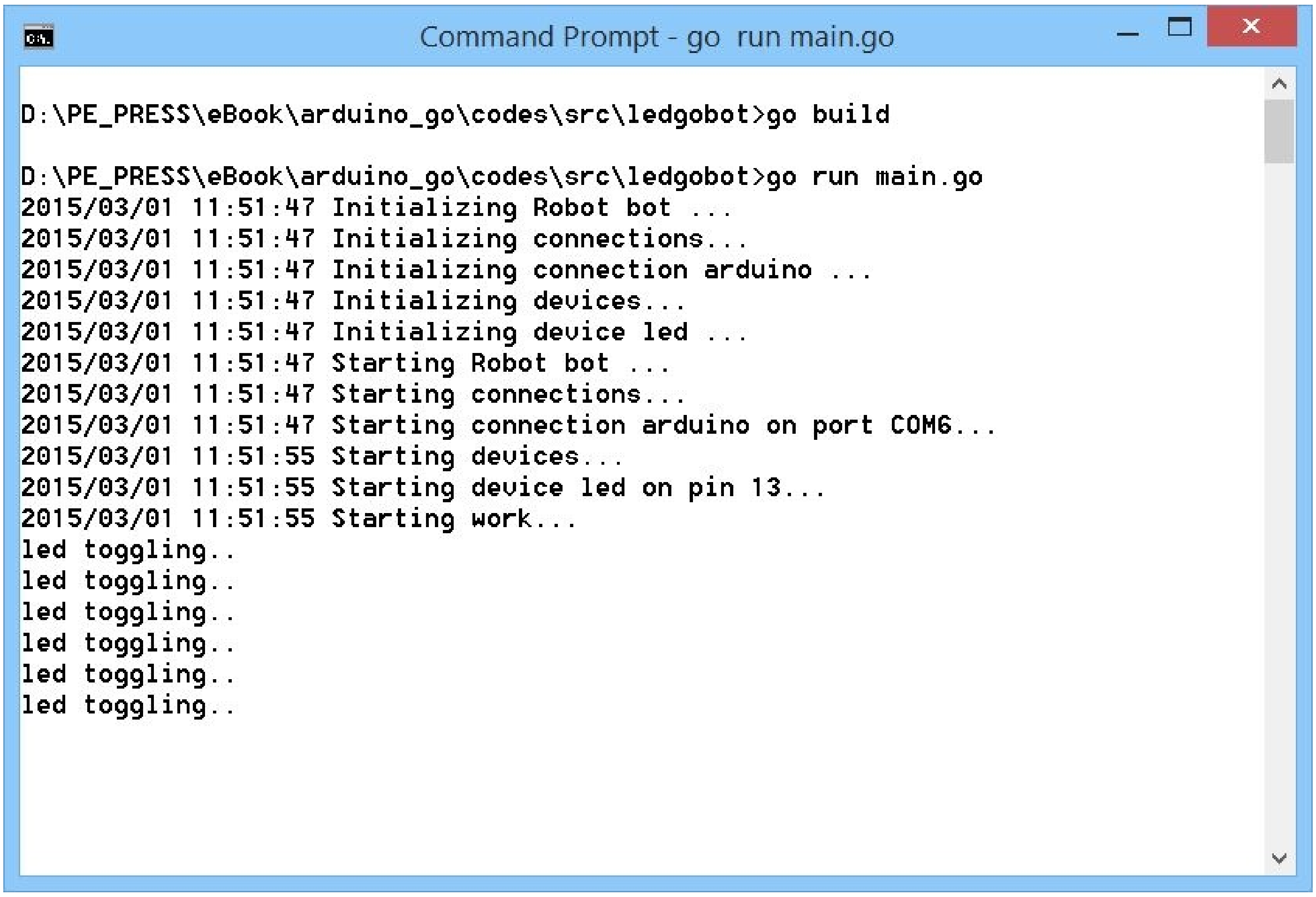
```
package main

import (
    "fmt"
    "time"
    "github.com/hybridgroup/gobot"
    "github.com/hybridgroup/gobot/platforms/firmata"
    "github.com/hybridgroup/gobot/platforms/gpio"
)

func main() {
    gbot := gobot.NewGobot()

    // change Arduino port, windows COMx. Linux /dev/ttyACMx
    firmataAdaptor := firmata.NewFirmataAdaptor("arduino", "COM6")
    led := gpio.NewLedDriver(firmataAdaptor, "led", "13")

    // function to run led blinking
    work := func() {
        gobot.Every(1*time.Second, func() {
            fmt.Println("led toggling..")
            led.Toggle()
        })
    }

    gbot.AddRobot(gobot.NewRobot("bot",
        []gobot.Connection{firmataAdaptor},
        []gobot.Device{led},
        work,
    ))

    gbot.Start()
}
```

Save this code and run it.

Now you can build and run this program. Don’t forget to connect your Arduino into PC via USB.

```
$ go build
$ go run main.go
```

On console, you can see the output as below.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\ledgobot>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\ledgobot>go run main.go
2015/03/01 11:51:47 Initializing Robot bot ...
2015/03/01 11:51:47 Initializing connections...
2015/03/01 11:51:47 Initializing connection arduino ...
2015/03/01 11:51:47 Initializing devices...
2015/03/01 11:51:47 Initializing device led ...
2015/03/01 11:51:47 Starting Robot bot ...
2015/03/01 11:51:47 Starting connections...
2015/03/01 11:51:47 Starting connection arduino on port COM6...
2015/03/01 11:51:55 Starting devices...
2015/03/01 11:51:55 Starting device led on pin 13...
2015/03/01 11:51:55 Starting work...
led toggling..
led toggling..
led toggling..
led toggling..
led toggling..
led toggling..
```

Then, you can see LED is blinking on the Arduino board.

DIGITAL(PWM ~)
TX
RX
L
ARDUINO
UNO
ON
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

3.3 go-firmata

The second Firmata package for Go is go-firmata. You can read it on https://github.com/kraman/go-firmata . You can install this package by typing this command.

```
$ go get github.com/kraman/go-firmata
```

For illustration, we build a simple app, blinking led. We use LED on the board, LED on pin 13.

Now we create project called firmataclient by creating a folder, firmataclient.

```
$ mkdir firmataclient
$ cd firmataclient
```

Create a file, called **main.go**. Write the following code.

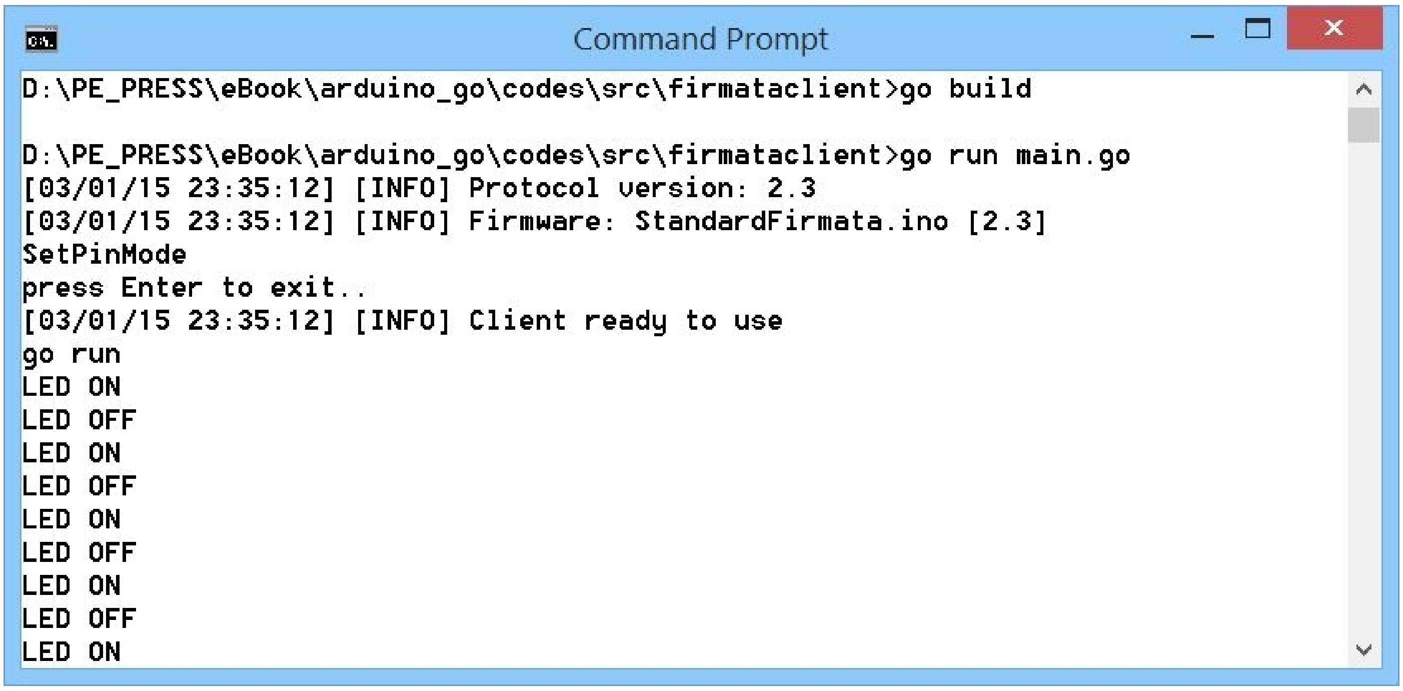
```
package main

import (
    "fmt"
    "time"
    "github.com/kraman/go-firmata"

)

func main() {
    board, err := firmata.NewClient("COM6",57600)
    if err!=nil {
        panic(err)
    }
    defer board.Close()
    fmt.Println("SetPinMode")
    board.SetPinMode(13,firmata.Output)

    go func() {
        fmt.Println("go run")
        for  {
            fmt.Println("LED ON")
            board.DigitalWrite(13,true)
            time.Sleep(1 * time.Second)
            fmt.Println("LED OFF")
            board.DigitalWrite(13,false)
            time.Sleep(1 * time.Second)
        }
    }()

    // press ENTER to exit
    fmt.Println("press Enter to exit..")
    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}
```

Save this code and run it.

Now you can build and run this program. Don’t forget to connect your Arduino into PC via USB.

```
$ go build
$ go run main.go
```

On console, you can see the output as below.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\firmataclient>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\firmataclient>go run main.go
[03/01/15 23:35:12] [INFO] Protocol version: 2.3
[03/01/15 23:35:12] [INFO] Firmware: StandardFirmata.ino [2.3]
SetPinMode
press Enter to exit..
[03/01/15 23:35:12] [INFO] Client ready to use
go run
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
```

3.4 Reading Digital Input

After created a blinking led, we can continue to build app which reads digital input. We need a switch/button and a LED. Connect your digital switch on Arduino digital pin 6 and LED on Arduino digital 9.

3.4.1 gobot

You create project called gobutton by creating a folder, gobutton.

```
$ mkdir gobutton
$ cd gobutton
```

Create a file, called **main.go**. Write the following code.

```
package main

import (
    "fmt"
    "time"
    "github.com/hybridgroup/gobot"
    "github.com/hybridgroup/gobot/platforms/firmata"
    "github.com/hybridgroup/gobot/platforms/gpio"
)

func main() {

    gbot := gobot.NewGobot()

    // change Arduino port, windows COMx. Linux /dev/ttyACMx
    firmataAdaptor := firmata.NewFirmataAdaptor("myFirmata", "COM6")

    // create a button on digital pin 6 and a led on digital pin 9
    button := gpio.NewButtonDriver(firmataAdaptor, "myButton", "6")
    led := gpio.NewLedDriver(firmataAdaptor, "led", "9")

    work := func() {
        gobot.On(button.Event("push"), func(data interface{}) {
            fmt.Println("button is pressed")
            led.On()
        })
        gobot.On(button.Event("release"), func(data interface{}) {
            fmt.Println("button is released")
            led.Off()
        })
    }

    gbot.AddRobot(gobot.NewRobot("bot",
        []gobot.Connection{firmataAdaptor},
        []gobot.Device{button,led},
        work,
    ))

    gbot.Start()

}
```

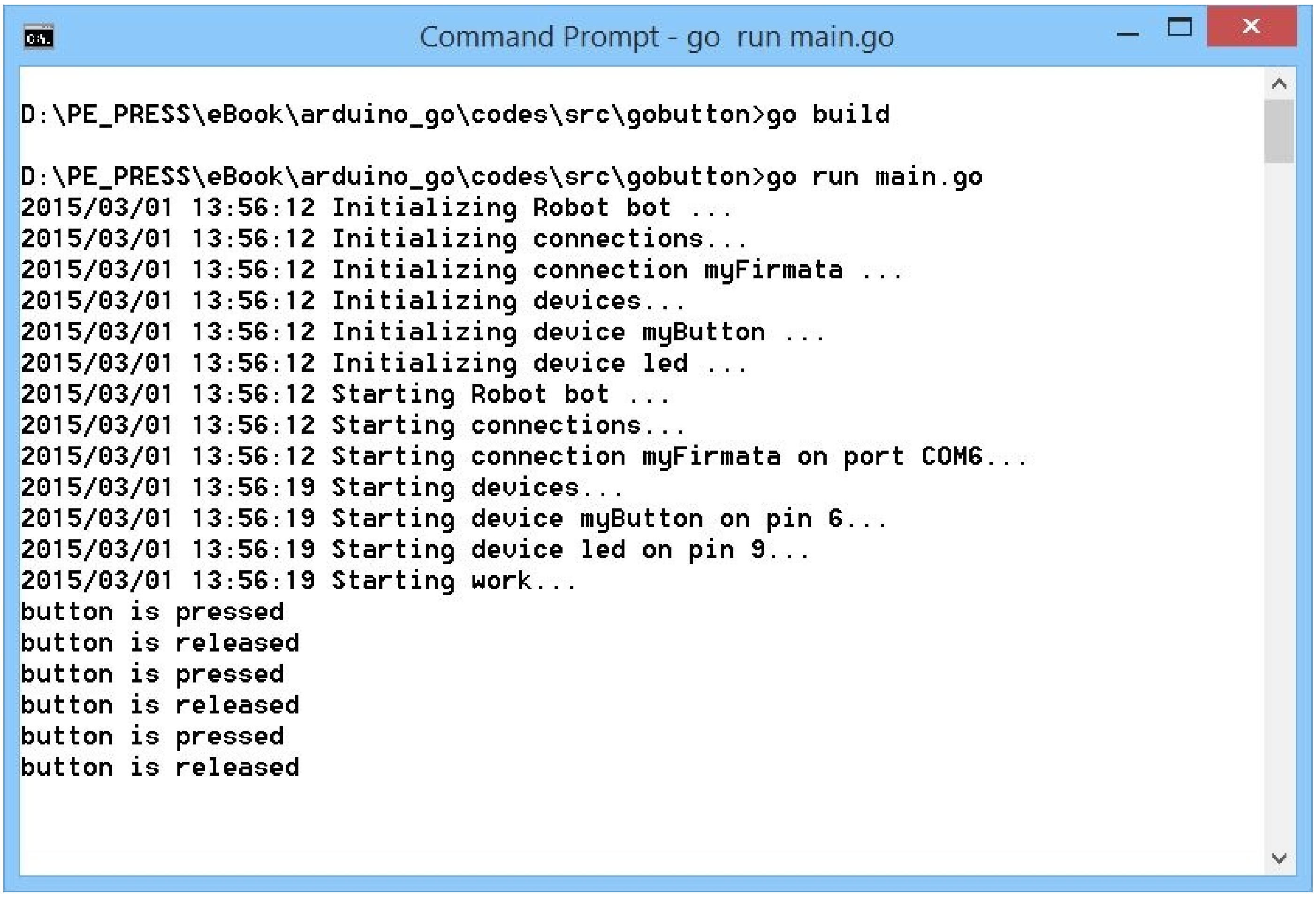
Save this code and run it.

Now you can build and run this program. Don't forget to connect your Arduino into PC via USB.

```
$ go build
$ go run main.go
```

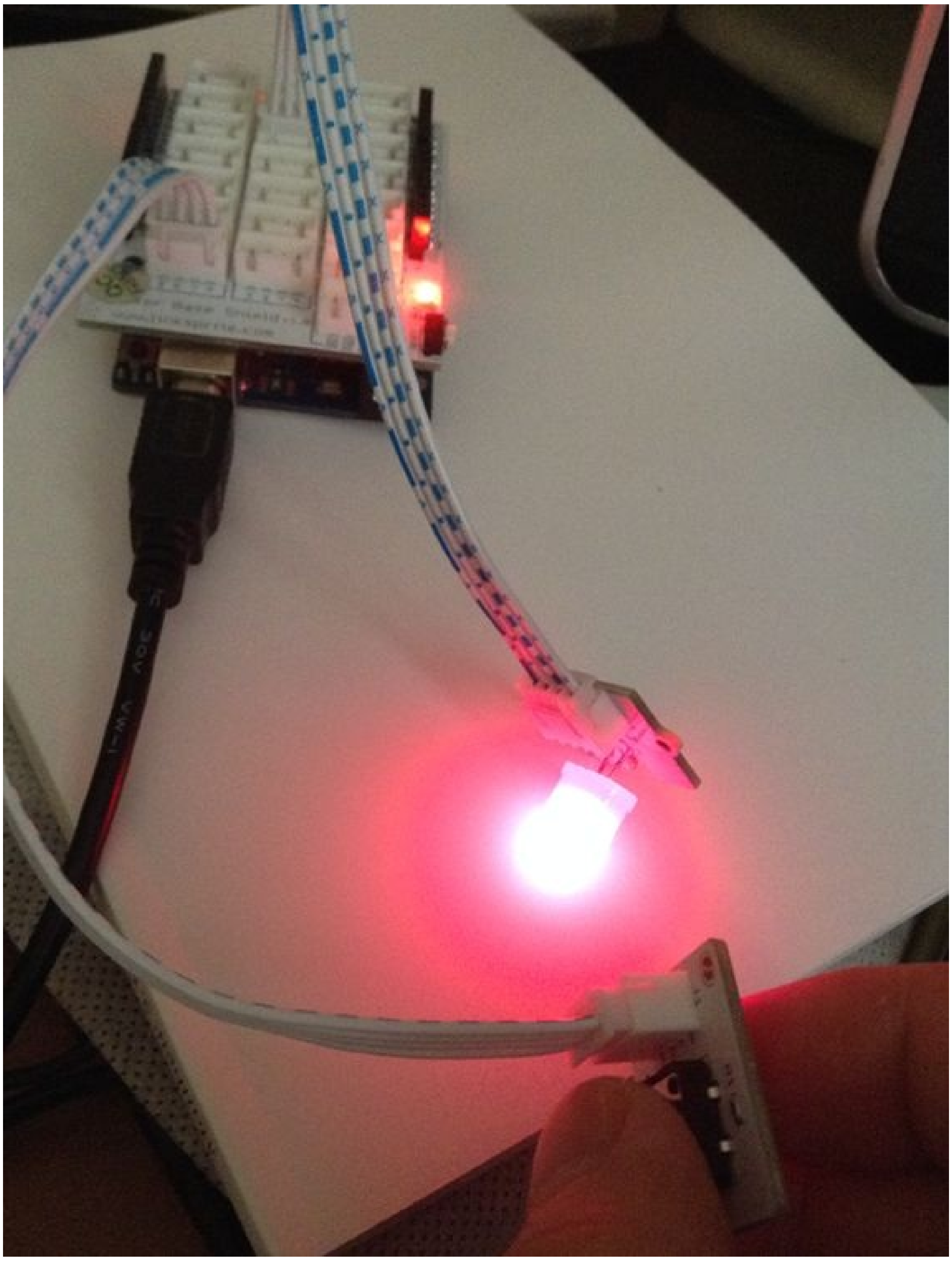
Press the button so you can see LED is ON.

On console, you can see the output as below.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\gobutton>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\gobutton>go run main.go
2015/03/01 13:56:12 Initializing Robot bot ...
2015/03/01 13:56:12 Initializing connections...
2015/03/01 13:56:12 Initializing connection myFirmata ...
2015/03/01 13:56:12 Initializing devices...
2015/03/01 13:56:12 Initializing device myButton ...
2015/03/01 13:56:12 Initializing device led ...
2015/03/01 13:56:12 Starting Robot bot ...
2015/03/01 13:56:12 Starting connections...
2015/03/01 13:56:12 Starting connection myFirmata on port COM6...
2015/03/01 13:56:19 Starting devices...
2015/03/01 13:56:19 Starting device myButton on pin 6...
2015/03/01 13:56:19 Starting device led on pin 9...
2015/03/01 13:56:19 Starting work...
button is pressed
button is released
button is pressed
button is released
button is pressed
button is released
```

A sample output on hardware schema.



3.4.2 go-firmata

Create project called firmatabutton by creating a folder, firmatabutton.

```
$ mkdir firmatabutton
$ cd firmatabutton
```

Create a file, called **main.go**. Write the following code.

```
package main
```

```go
import (
    "fmt"
    "time"
    "github.com/kraman/go-firmata"

)

func main() {
    board, err := firmata.NewClient("COM6",57600)
    if err!=nil {
        panic(err)
    }
    defer board.Close()
    // LED on digital pin 9
    board.SetPinMode(9,firmata.Output)
    // button on digital pin 6
    board.SetPinMode(6,firmata.Input)
    board.EnableDigitalInput(6,true)


    go func() {

        for buttonVal := range board.GetValues(){
            _,val,_ := firmata.FirmataValue(buttonVal).GetDigitalValue()

            if val[6]==true {
                fmt.Println("LED ON")
                board.DigitalWrite(9,true)
            }else{
                fmt.Println("LED OFF")
                board.DigitalWrite(9,false)
            }

            time.Sleep(700 * time.Millisecond)
        }

    }()

    // press ENTER to exit
    fmt.Println("press Enter to exit..")
    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}
```

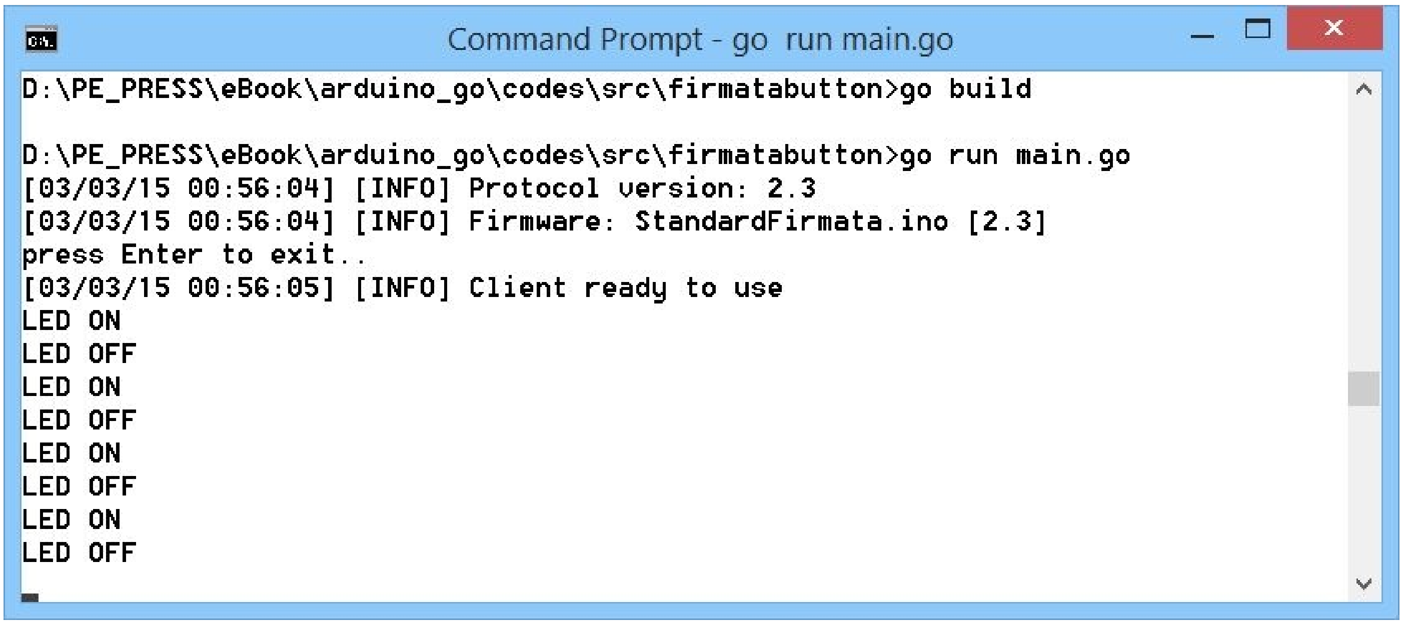
Save this code and run it.

Now you can build and run this program. Don’t forget to connect your Arduino into PC via USB.

```
$ go build
$ go run main.go
```

Press the button so you can see LED is ON.

On console, you can see the output as below.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\firmatabutton>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\firmatabutton>go run main.go
[03/03/15 00:56:04] [INFO] Protocol version: 2.3
[03/03/15 00:56:04] [INFO] Firmware: StandardFirmata.ino [2.3]
press Enter to exit..
[03/03/15 00:56:05] [INFO] Client ready to use
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
LED ON
LED OFF
```

4. Analog Sensor

In this chapter I’m going to explain how to access sensor data on Arduino with sensor device. In this section, I use go-firmata, https://github.com/kraman/go-firmata to develop Go application to access sensor device on Arduino.

4.1 Sensor Devices

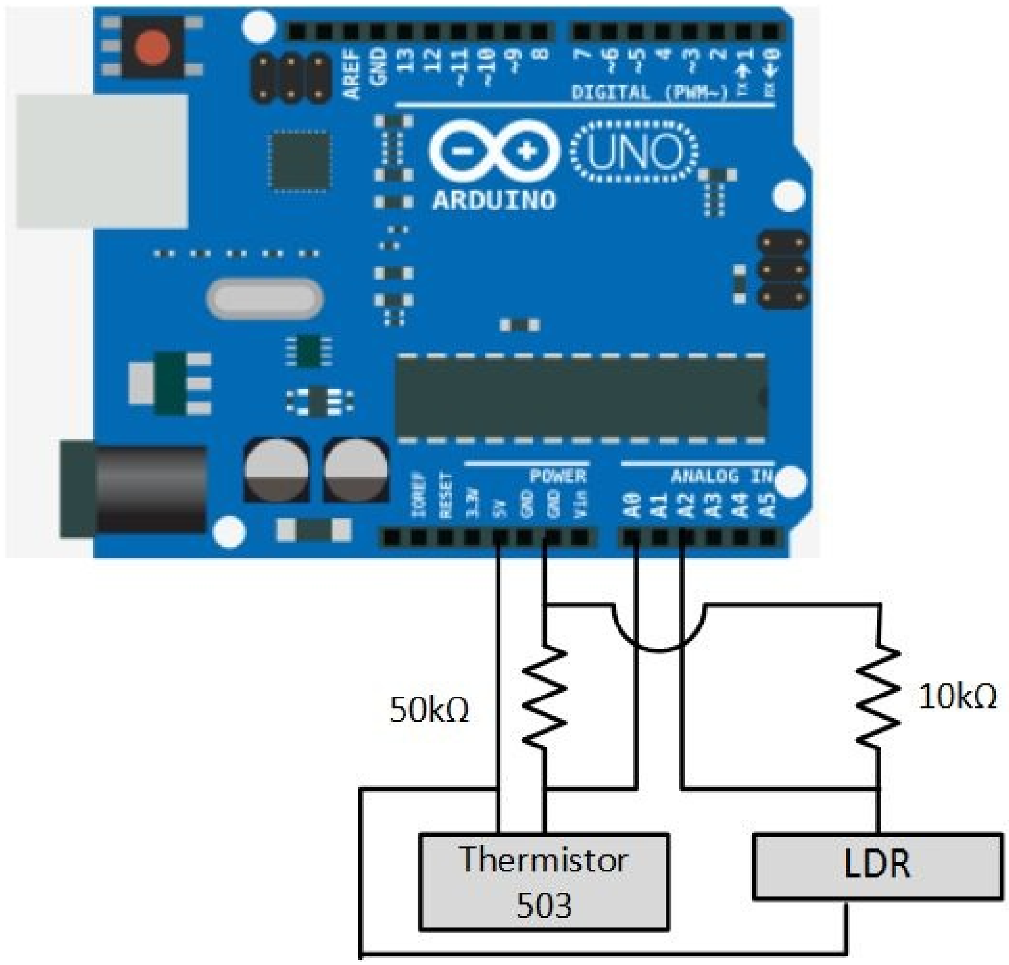
Arduino can be interfaced with sensor devices. You can see the list of sensor interface on http://playground.arduino.cc/Main/InterfacingWithHardware .

In this scenario, we use cheap sensor devices, Themistors and LDR (Light Dependent Resistors). Thermistor can be used to measure temperature and LDR can be used to measure light and dark based on illumination.

I have Thermistor 503 and LDR from Arduino Sidekick Basic kit.

For building sensor hardware, we need 2 resistors, 50k ohm and 10k ohm. If your thermistor is 10k ohm, you should use a resistor with 10k ohm.

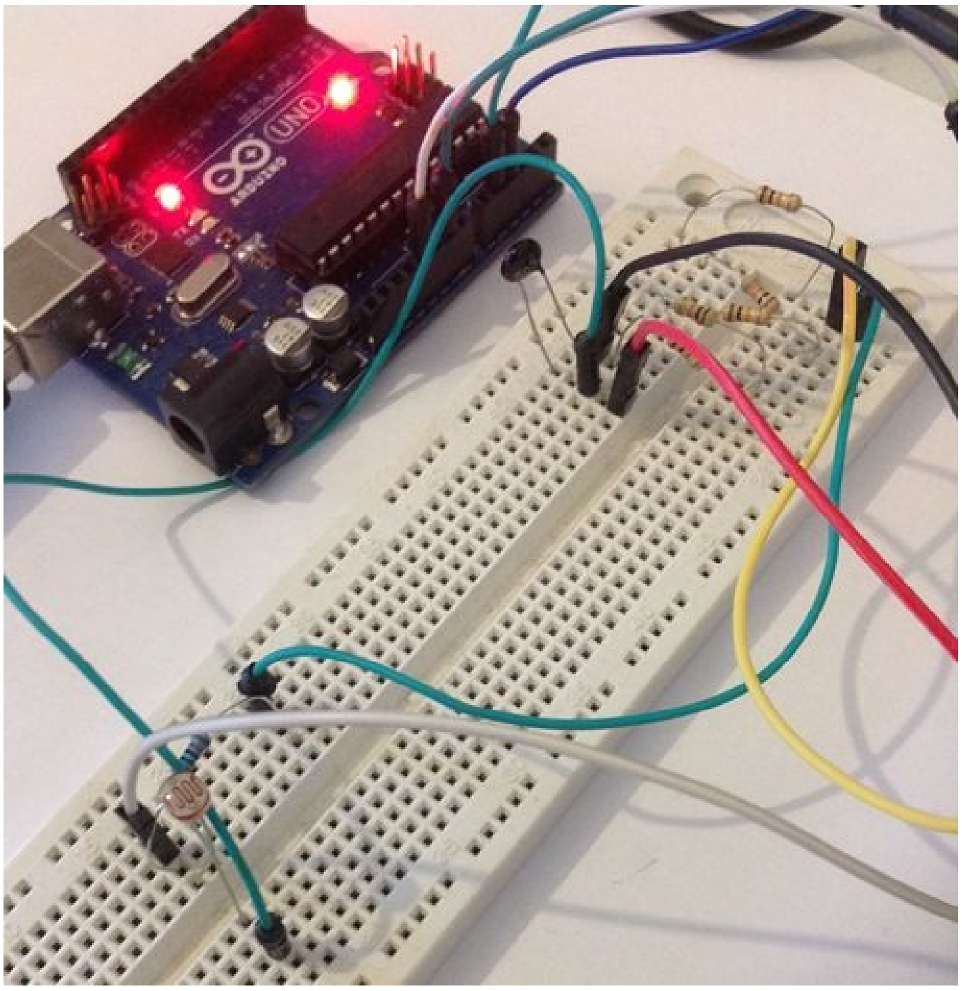
The following is a circuit schema for Thermistor and LDR sensors.



One of sensor pin is connected to 5V Arduino pin. We use a divider approach. Thermistor 503 is attached to the **Analog In** of Arduino, **A0**. Otherwise, LDR is attached to **A2**.

You may attach LED as the indicator for sensor reading state. You can use a solution on chapter 2, Blink app.

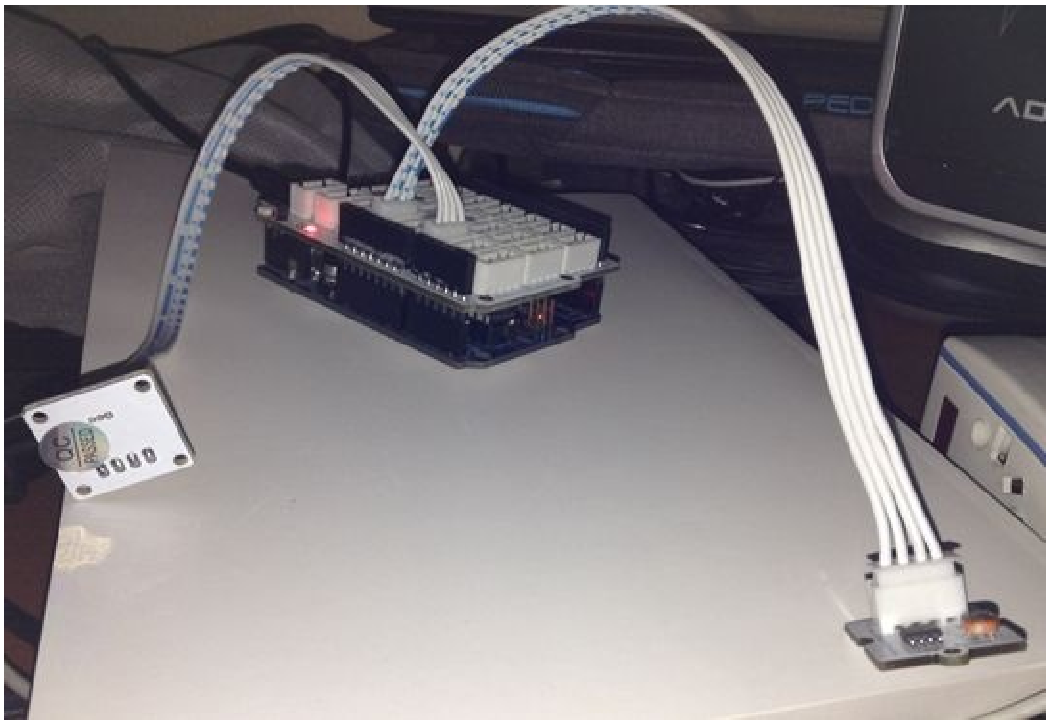
Here is a sample of hardware implementation.



You also can use Arduino shield which sensor devices are be connected, for instance, I use Linker kit from http://store.linksprite.com/linker-kit/ with the following sensor items:

- LDR, http://store.linksprite.com/ldr-ambient-light-module-of-linker-kit-for-pcduino-arduino/
- Temperature sensor, http://store.linksprite.com/thermal-module-of-linker-kit-for-pcduino-arduino/

Hardware implementation can be seen in Figure below.



4.2 Reading Sensor

How to read the sensor data from Arduino?. We use go-firmata, https://github.com/kraman/go-firmata. To access Arduino Analog from Firmata, we can convert the following pin:

- Arduino analog A0 –> Firmata pin 14
- Arduino analog A1 –> Firmata pin 15
- Arduino analog A2 –> Firmata pin 16
- Arduino analog A3 –> Firmata pin 17

Firstly, we create project called sensordemo by creating a folder, sensordemo.

```
$ mkdir sensordemo
$ cd sensordemo
```

Create a file, called **main.go**. Write the following code.

```
package main

import (
    "fmt"
    "math"
    "time"
    arduino "github.com/kraman/go-firmata"

)

func main() {
    board, err := arduino.NewClient("COM6",57600)
    if err!=nil {
        panic(err)
    }
    defer board.Close()

    // LDR on A0 = Pin 14
    err1 := board.EnableAnalogInput(14,true)
    if err1!=nil {
        panic(err1)
    }
    // Temperature on A2 = Pin 16
    err2 := board.EnableAnalogInput(16,true)
    if err2!=nil {
        panic(err2)
    }
    go func() {

        for  {
            for sensorVal := range board.GetValues(){
                pin,val,_ := arduino.FirmataValue(sensorVal).GetAnalogVa
                fmt.Println(pin,":",val)

                if pin==16 {
                    // linker kit
                    temperatureC := CalculateTempLinkerKit(float64(val))
                    // Thermistor
                    //temperatureC := CalculateThermistor(float64(val))
                    fmt.Println("Temperature:",temperatureC, "degrees C"
                }

                if pin == 14 {

                    // if you use LDR
                    //ldr := val
                    // if you use LDR from linker kit
                    ldr := CalculateLDR(float64(val))
                    if ldr < 10 {
                        fmt.Println("LDR:",ldr, "Dark")

                    }else if ldr <200 {
                        fmt.Println("LDR:",ldr, "Dim")

                    }else if ldr <500 {
                        fmt.Println("LDR:",ldr, "Light")

                    }else if ldr <800 {
                        fmt.Println("LDR:",ldr, "Bright")

                    }else{
                        fmt.Println("LDR:",ldr, "Very bright")
                    }

                }
                time.Sleep(1000 * time.Millisecond)
            }

            time.Sleep(1 * time.Second)
        }
    }()

    // press ENTER to exit
    fmt.Println("press Enter to exit..")
    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}

// based on http://playground.arduino.cc/ComponentLib/Thermistor2
func CalculateThermistor(rawADC float64) float64 {
    temp := math.Log(((10240000 / rawADC) - 10000))
    temp = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * temp * t
    temp -= 273.15     // Convert Kelvin to Celsius
    return temp

}
// LDR from linker kit
// http://store.linksprite.com/ldr-ambient-light-module-of-linker-kit-fo
func CalculateLDR(sensorValue float64) float64 {
    val := float64(1023-sensorValue)*10/sensorValue;

    return val
}
// Temperature sensor from linker kit
func CalculateTempLinkerKit(rowADC float64) float64{
    voltage := float64(rowADC) * 5.0
    voltage /= 1024.0;
    temperatureC := (voltage - 0.5) * 100

    return temperatureC

}
```

This code works for LDR & Thermistor and Linker kit. You can remark one of them.

Save this code and run it.

You can see that LDR sensor value is converted to light information, for instance dark, dim, light, bright, very bright.

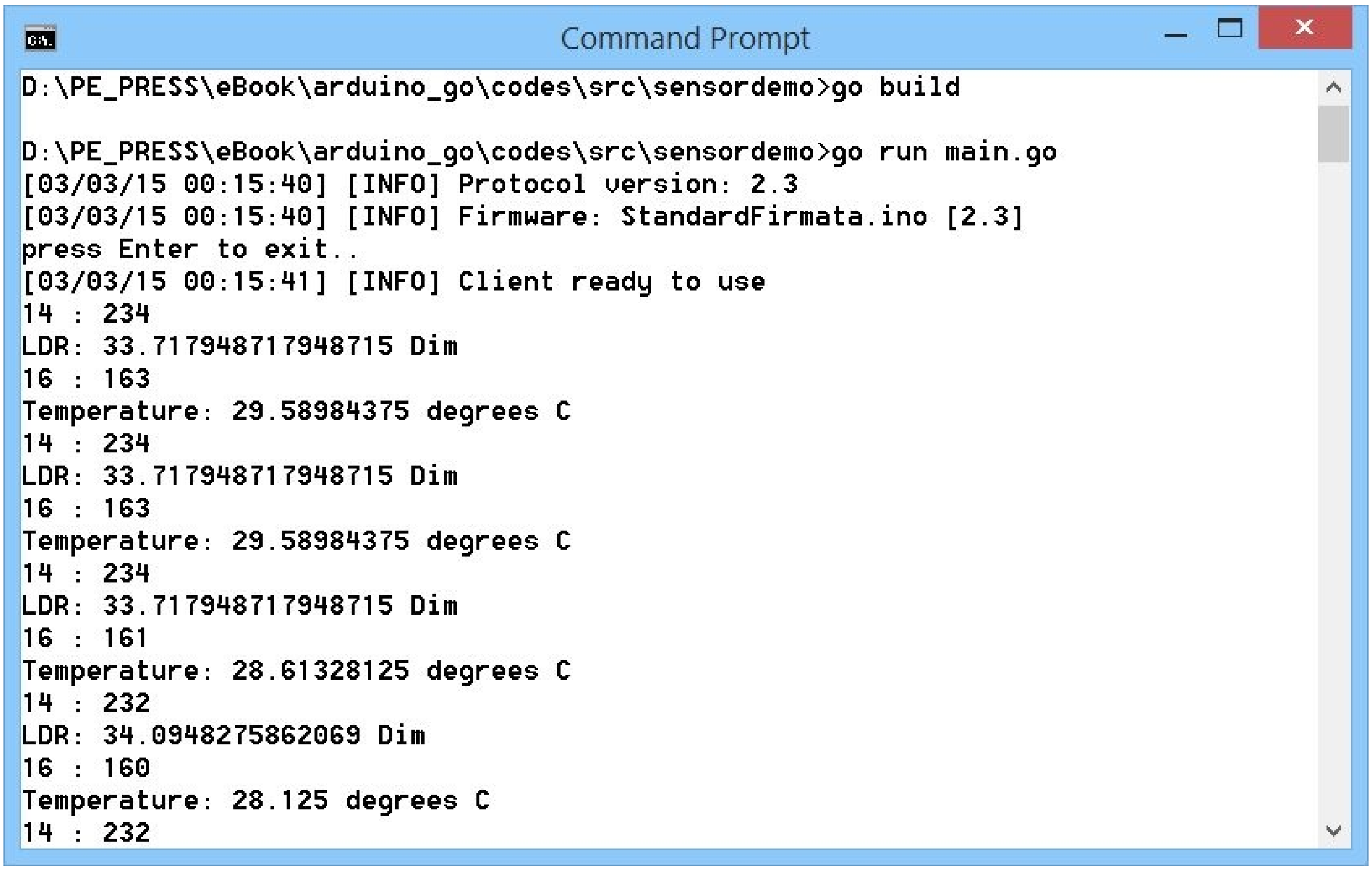
Save this code.

4.3 Running Program

Now you can execute your program. Type this command.

```
$ go build
$ go run main.go
```

You can see a sample output as below.



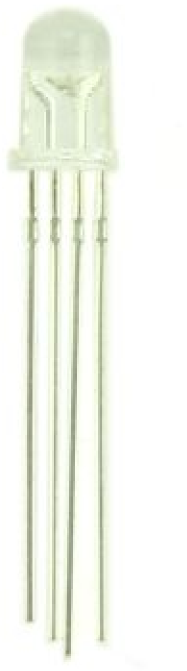
```
D:\PE_PRESS\eBook\arduino_go\codes\src\sensordemo>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\sensordemo>go run main.go
[03/03/15 00:15:40] [INFO] Protocol version: 2.3
[03/03/15 00:15:40] [INFO] Firmware: StandardFirmata.ino [2.3]
press Enter to exit..
[03/03/15 00:15:41] [INFO] Client ready to use
14 : 234
LDR: 33.717948717948715 Dim
16 : 163
Temperature: 29.58984375 degrees C
14 : 234
LDR: 33.717948717948715 Dim
16 : 163
Temperature: 29.58984375 degrees C
14 : 234
LDR: 33.717948717948715 Dim
16 : 161
Temperature: 28.61328125 degrees C
14 : 232
LDR: 34.0948275862069 Dim
16 : 160
Temperature: 28.125 degrees C
14 : 232
```

5. RGB LED

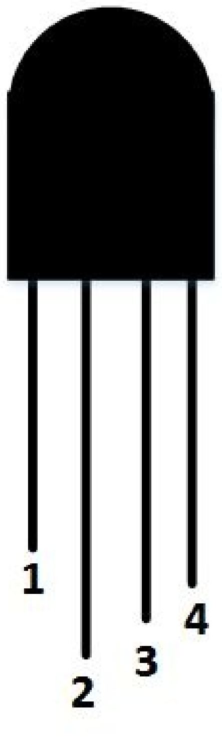
This chapter explains how to control RGB LED connected to Arduino board using Go. We explore how to access PWM (Pulse Width Modulation) Arduino.

5.1 RGB LED

In this scenario we build a Go application to control RGB LED color using Arduino Analog output (PWM). RGB LED has 4 pins that you can see it on Figure below.



To understand these pins, you can see the following Figure.



Note:

- Pin 1: Red
- Pin 2: Common pin

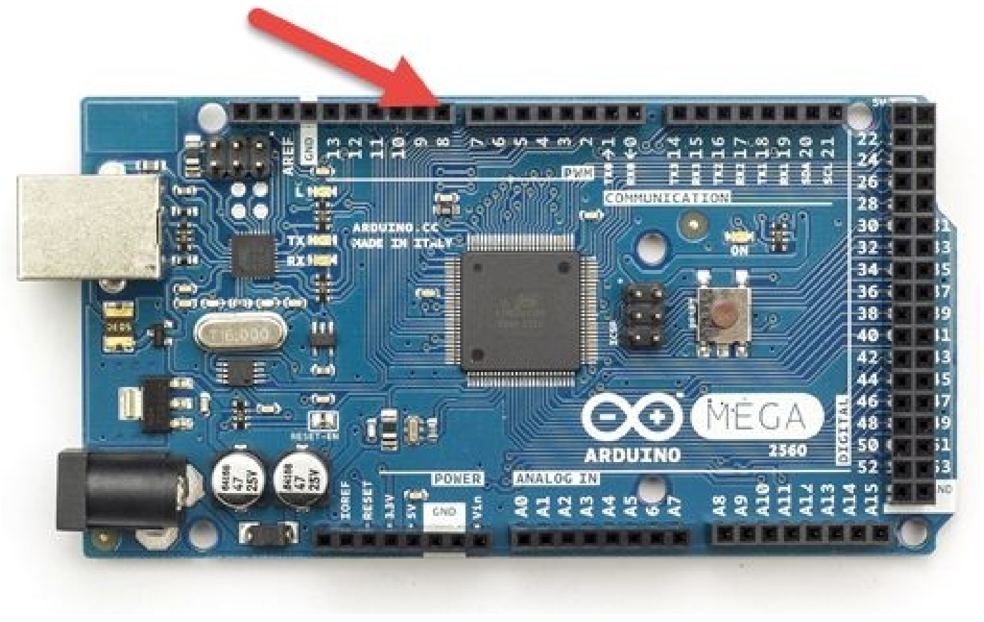
- Pin 3: Green
- Pin 4: Blue

Now we can start to build a Go application and hardware implementation.

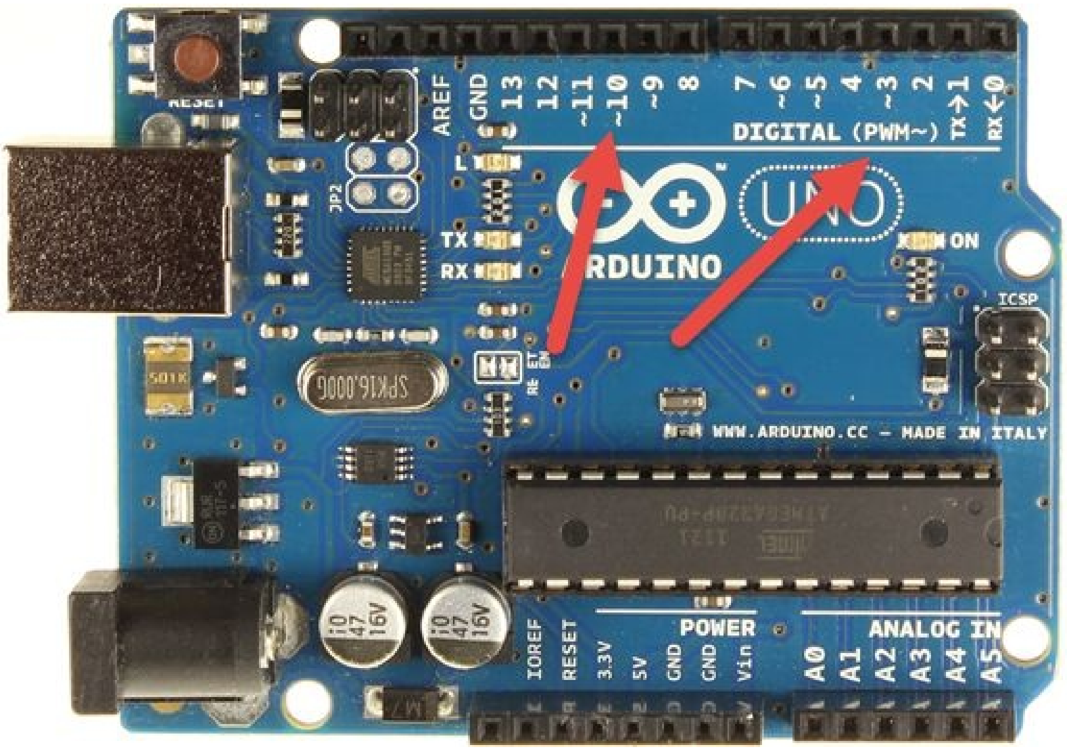
5.1.1 Arduino Analog output (PWM)

Please be careful if you want to work with Arduino PWM. If you have Arduino Mega, you will see PWM label so you obtain PWM pins easily but if you have Arduino Uno, it writes DIGITAL (PWM ~). It means your PWM pins can be found on DIGITAL pins which pin with ~, for instance, ~3,~5,~6,~9, ~10, ~11.

For Arduino Mega 2560, you can see PWM pins on picture below (see red arrow).

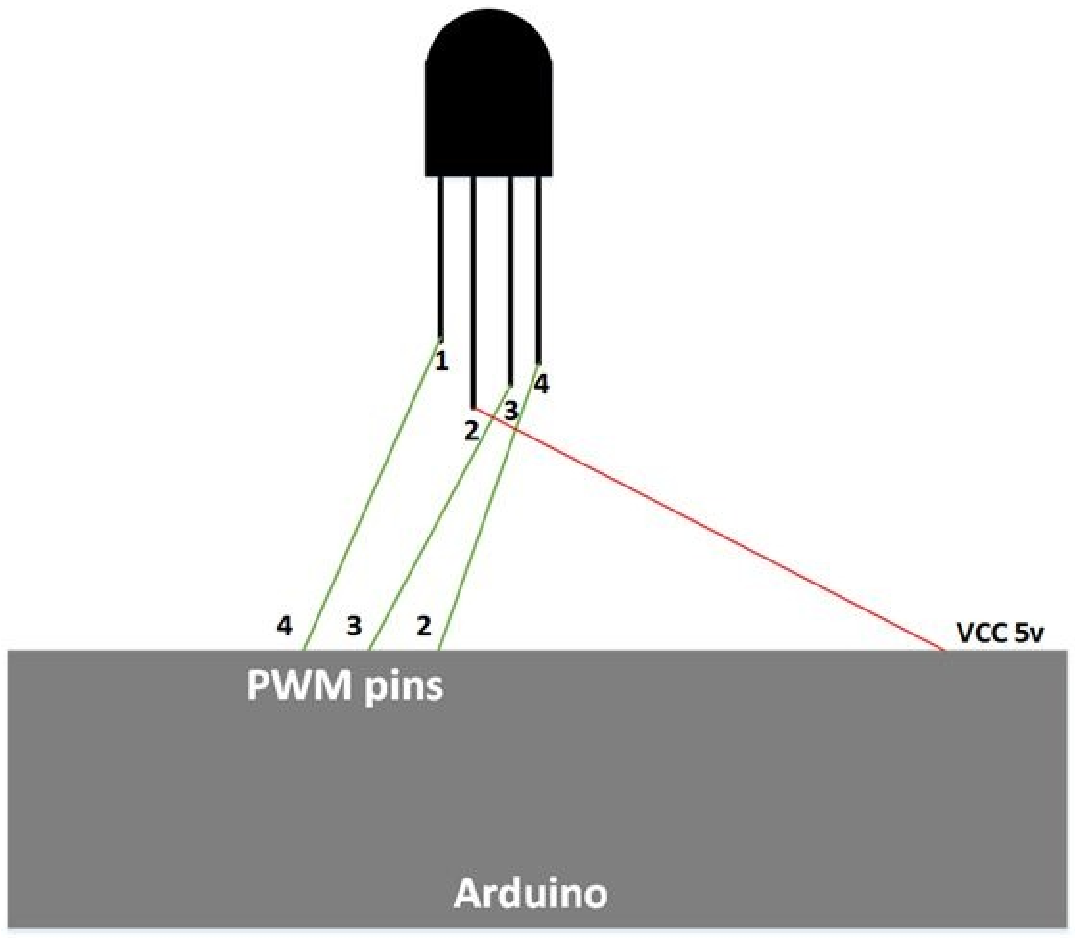


For Arduino Uno R3, you can see PWM pins as below.



5.1.2 Controlling RGB LED Color

Firstly we implement RGB LED hardware. The following is a hardware schema.



For our testing, we configure the following PWM pins.

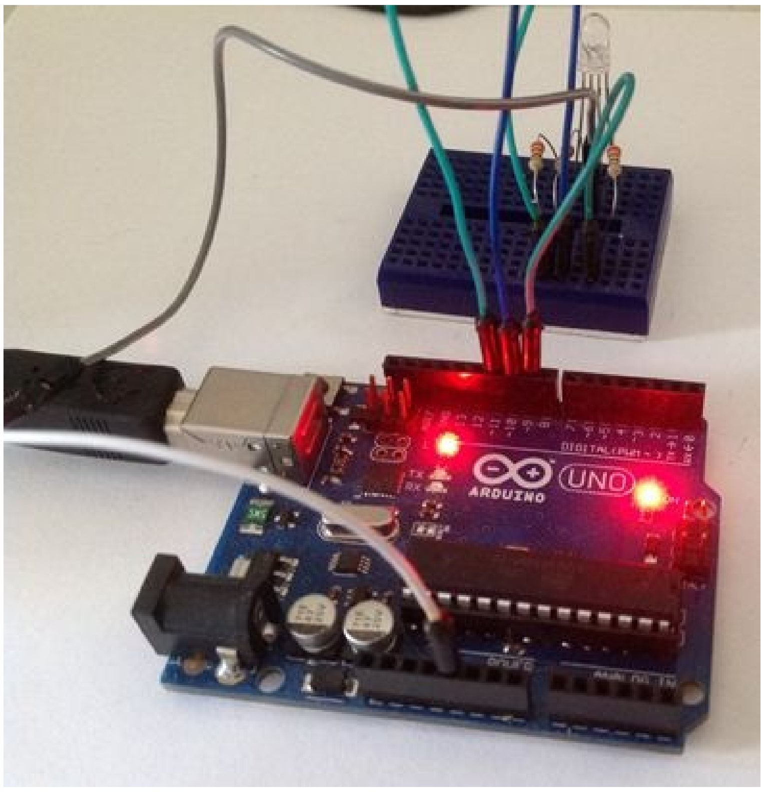
Arduino Mega 2560:

- RGB LED pin 1 (red) is connected to Arduino PWM pin 4
- RGB LED pin 2 is connected to Arduino VCC 5V
- RGB LED pin 3 (green) is connected to Arduino PWM pin 3
- RGB LED pin 4 (blue) is connected to Arduino PWM pin 2

Arduino Uno R3:

- RGB LED pin 1 (red) is connected to Arduino PWM pin 9
- RGB LED pin 2 is connected to Arduino VCC 5V
- RGB LED pin 3 (green) is connected to Arduino PWM pin 10
- RGB LED pin 4 (blue) is connected to Arduino PWM pin 11

Here is a sample implementation with Arduino Uno R3.



5.2 Arduino Implementation

Now we implement our RGB LED controller in Arduino. This is for testing. Open Firstly, we define our RGB LED pins. The following is RGB LED pins for Arduino Mega 2560.

```
int redPin = 4;
int greenPin = 3;
int bluePin = 2;
```

We can define RGB LED pins for Arduino Uno R3.

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;
```

Now we initialize pins on setup().

```
void setup()
{
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
    Serial.begin(9600);
}
```

We define a function, called setColor(). This function aims to write RGB values on PWM pins.

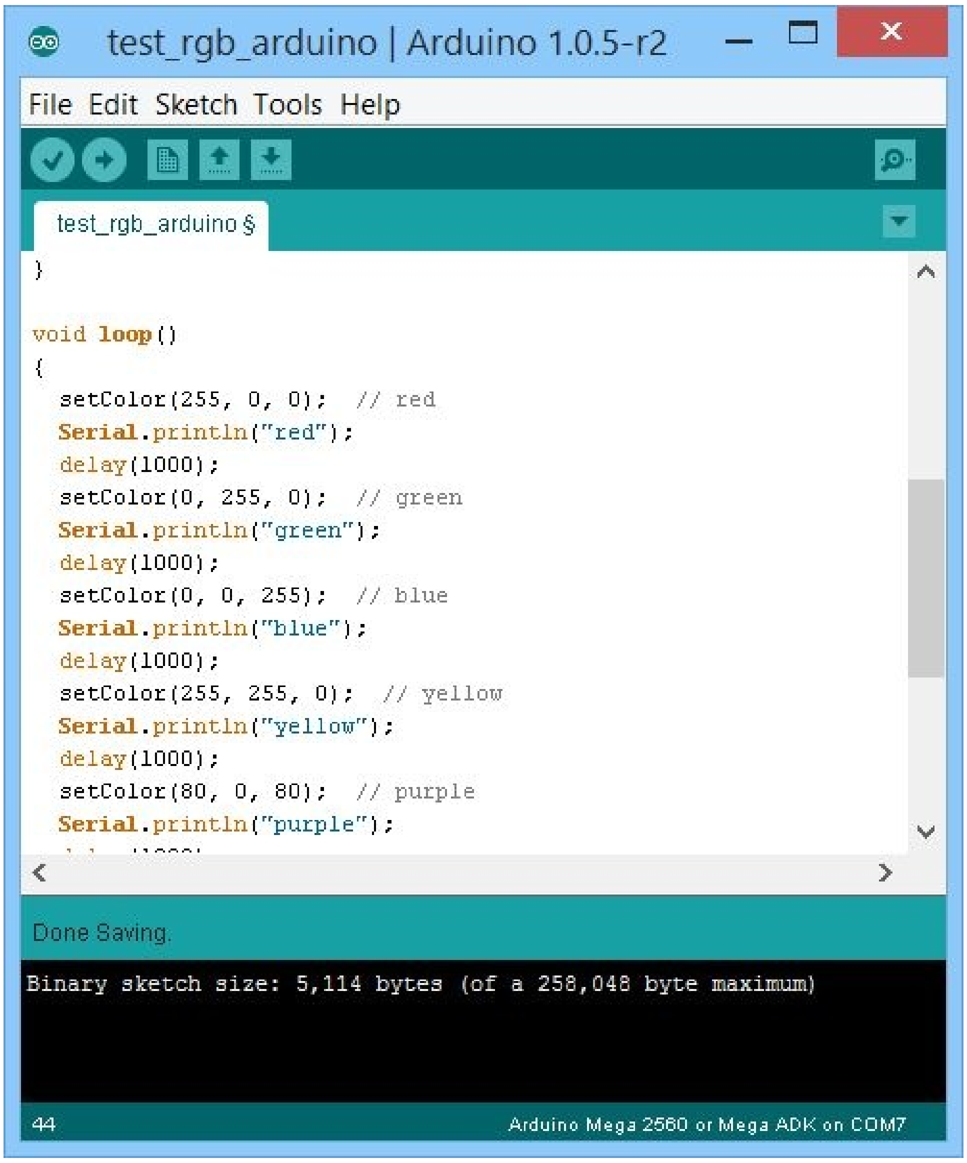
```
void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Now we control RGB values on RGB LED, for instance, Red, Green, Blue, Yellow, Purple, Aqua.

```
void loop()
{
  setColor(255, 0, 0);  // red
  Serial.println("red");
  delay(1000);
  setColor(0, 255, 0);  // green
  Serial.println("green");
  delay(1000);
  setColor(0, 0, 255);  // blue
```

```
    Serial.println("blue");
    delay(1000);
    setColor(255, 255, 0);  // yellow
    Serial.println("yellow");
    delay(1000);
    setColor(80, 0, 80);  // purple
    Serial.println("purple");
    delay(1000);
    setColor(0, 255, 255);  // aqua
    Serial.println("aqua");
    delay(1000);
  }
```

Save this code, called **test_rgb_arduino.ino**.


test_rgb_arduino | Arduino 1.0.5-r2
File Edit Sketch Tools Help
test_rgb_arduino §

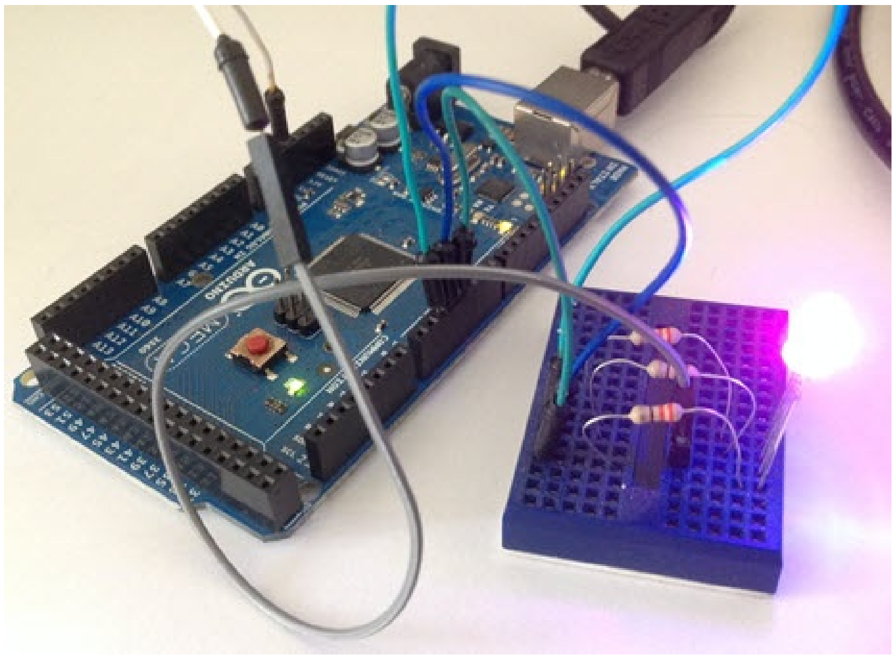
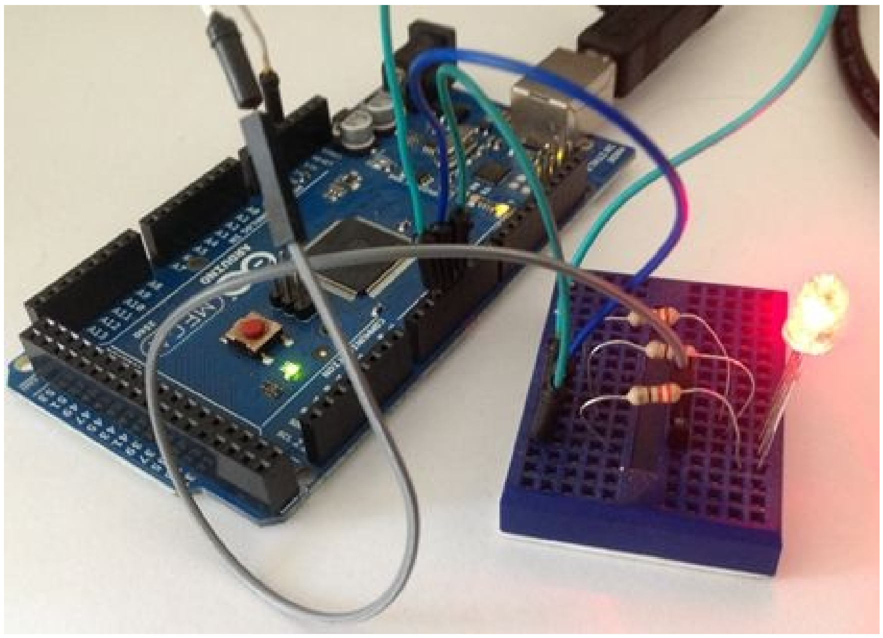
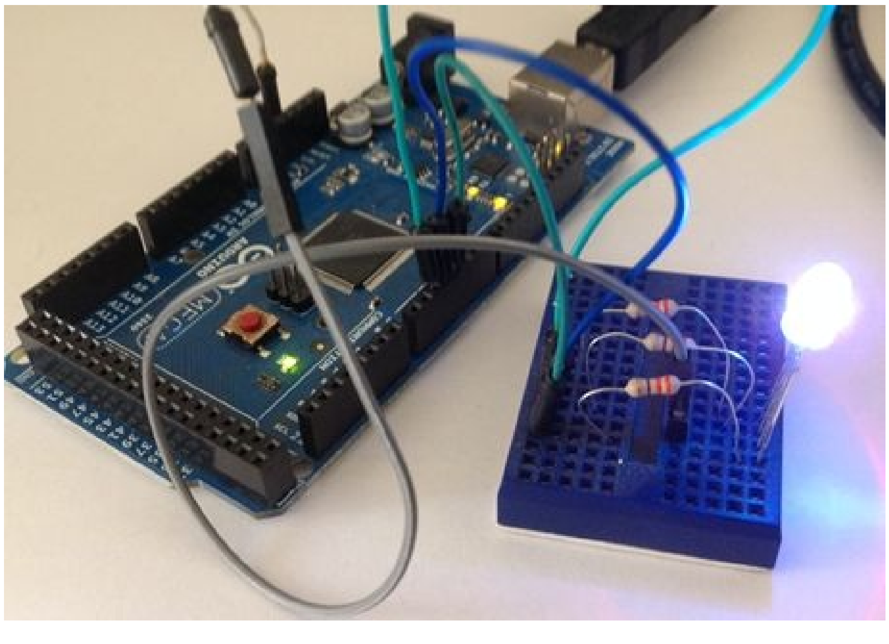
```
}

void loop()
{
  setColor(255, 0, 0);  // red
  Serial.println("red");
  delay(1000);
  setColor(0, 255, 0);  // green
  Serial.println("green");
  delay(1000);
  setColor(0, 0, 255);  // blue
  Serial.println("blue");
  delay(1000);
  setColor(255, 255, 0);  // yellow
  Serial.println("yellow");
  delay(1000);
  setColor(80, 0, 80);  // purple
  Serial.println("purple");
```

```
Done Saving.
Binary sketch size: 5,114 bytes (of a 258,048 byte maximum)
```

44
Arduino Mega 2560 or Mega ADK on COM7

Compile and verify this code. If success, you can upload it to Arduino board.

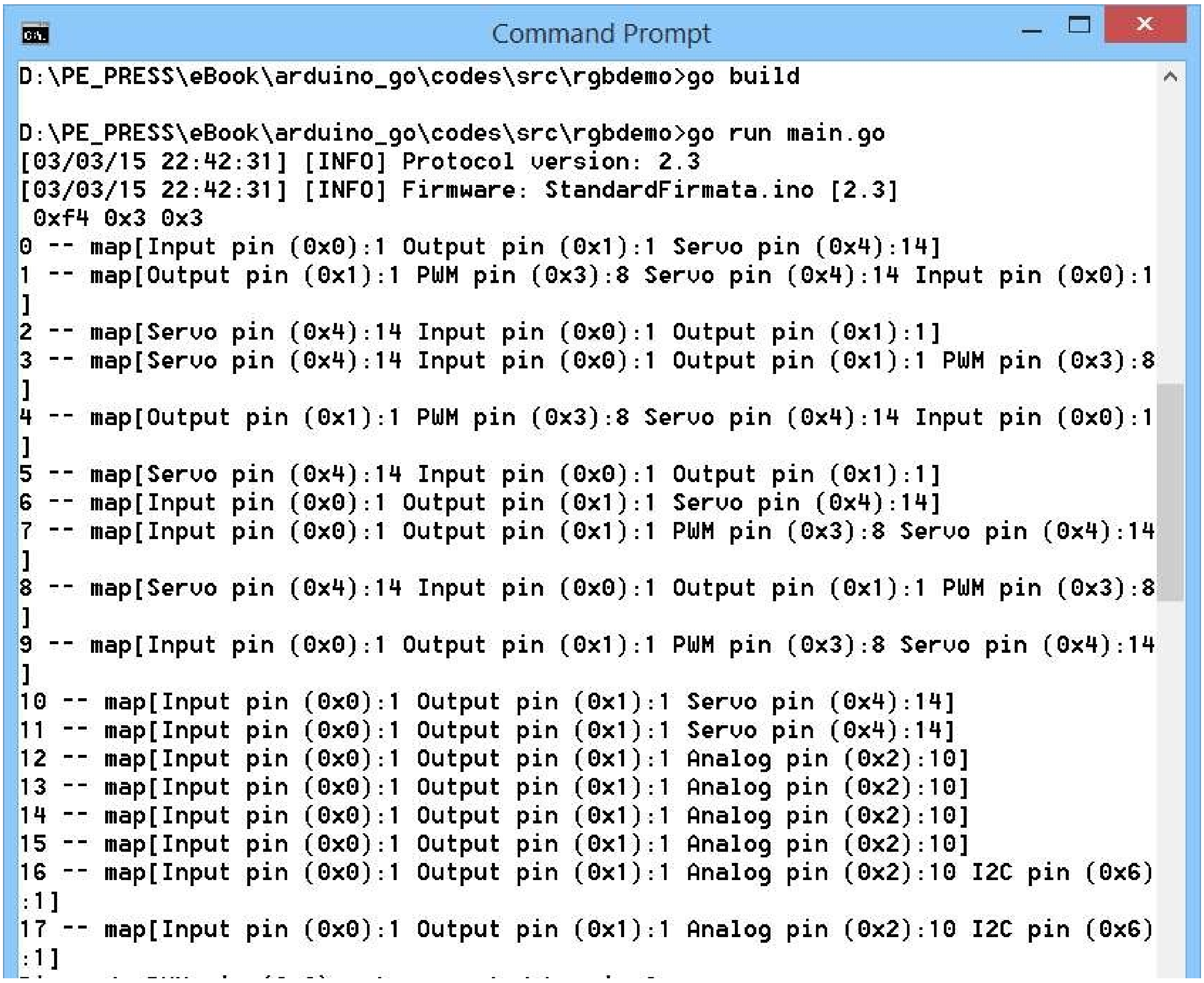
If success, you can see RGB LED blinking with different colors. Here is a sample output of RGB LED with Arduino Mega 2560.

5.3 Go Implementation

In this section, we create Go application to control RGB LED color. We use the same scenario as previous section. For testing, I used go-firmata package, https://github.com/kraman/go-firmata . Don't forget to load standard Firmata library on your Arduino. Further information, you read it on section 3.1.

I have tested PWM on pin 9,10 and 11 didn't work. After printed all pin mapping, I have the following pin mapping.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\rgbdemo>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\rgbdemo>go run main.go
[03/03/15 22:42:31] [INFO] Protocol version: 2.3
[03/03/15 22:42:31] [INFO] Firmware: StandardFirmata.ino [2.3]
 0xf4 0x3 0x3
0 -- map[Input pin (0x0):1 Output pin (0x1):1 Servo pin (0x4):14]
1 -- map[Output pin (0x1):1 PWM pin (0x3):8 Servo pin (0x4):14 Input pin (0x0):1
]
2 -- map[Servo pin (0x4):14 Input pin (0x0):1 Output pin (0x1):1]
3 -- map[Servo pin (0x4):14 Input pin (0x0):1 Output pin (0x1):1 PWM pin (0x3):8
]
4 -- map[Output pin (0x1):1 PWM pin (0x3):8 Servo pin (0x4):14 Input pin (0x0):1
]
5 -- map[Servo pin (0x4):14 Input pin (0x0):1 Output pin (0x1):1]
6 -- map[Input pin (0x0):1 Output pin (0x1):1 Servo pin (0x4):14]
7 -- map[Input pin (0x0):1 Output pin (0x1):1 PWM pin (0x3):8 Servo pin (0x4):14
]
8 -- map[Servo pin (0x4):14 Input pin (0x0):1 Output pin (0x1):1 PWM pin (0x3):8
]
9 -- map[Input pin (0x0):1 Output pin (0x1):1 PWM pin (0x3):8 Servo pin (0x4):14
]
10 -- map[Input pin (0x0):1 Output pin (0x1):1 Servo pin (0x4):14]
11 -- map[Input pin (0x0):1 Output pin (0x1):1 Servo pin (0x4):14]
12 -- map[Input pin (0x0):1 Output pin (0x1):1 Analog pin (0x2):10]
13 -- map[Input pin (0x0):1 Output pin (0x1):1 Analog pin (0x2):10]
14 -- map[Input pin (0x0):1 Output pin (0x1):1 Analog pin (0x2):10]
15 -- map[Input pin (0x0):1 Output pin (0x1):1 Analog pin (0x2):10]
16 -- map[Input pin (0x0):1 Output pin (0x1):1 Analog pin (0x2):10 I2C pin (0x6)
:1]
17 -- map[Input pin (0x0):1 Output pin (0x1):1 Analog pin (0x2):10 I2C pin (0x6)
:1]
```

You print your pin mapping by modifying code client.go from go-firmata package as follows.

```
func (c *FirmataClient) SetPinMode(pin byte, mode PinMode) (err error) {
  if c.pinModes[pin][mode] == nil {

    err = fmt.Errorf("Pin mode %v not supported by pin %v", mode, pin)
    return
  }
    // print all pin mapping
    for k, v := range c.pinModes {

        fmt.Println(k,"--",v)


    }
  cmd := []byte{byte(SetPinMode), (pin & 0x7F), byte(mode)}
  err = c.sendCommand(cmd)
  return
}
```

Then, test it.

From this case, I connect RGB LED on Arduino pin1 for RGB red, Arduino pin 3 for RGB green and Arduino pin 9 for RGB blue.

Now we create project called rgbdemo by creating a folder, rgbdemo.

```
$ mkdir rgbdemo
$ cd rgbdemo
```

Create a file, called **main.go**. Write the following code.

```
package main

import (
    "fmt"
    "time"
    arduino "github.com/kraman/go-firmata"

)

func main() {
    board, err := arduino.NewClient("COM6",57600)
    if err!=nil {
        panic(err)
    }
    defer board.Close()

    // define RGB pin
    board.SetPinMode(1,arduino.PWM)  // red
    board.SetPinMode(3,arduino.PWM) // green
    board.SetPinMode(9,arduino.PWM) // blue

    go func() {

        for  {

            SetColor(board,255, 0, 0)  // red
            fmt.Println("red")
            time.Sleep(2 * time.Second)
```

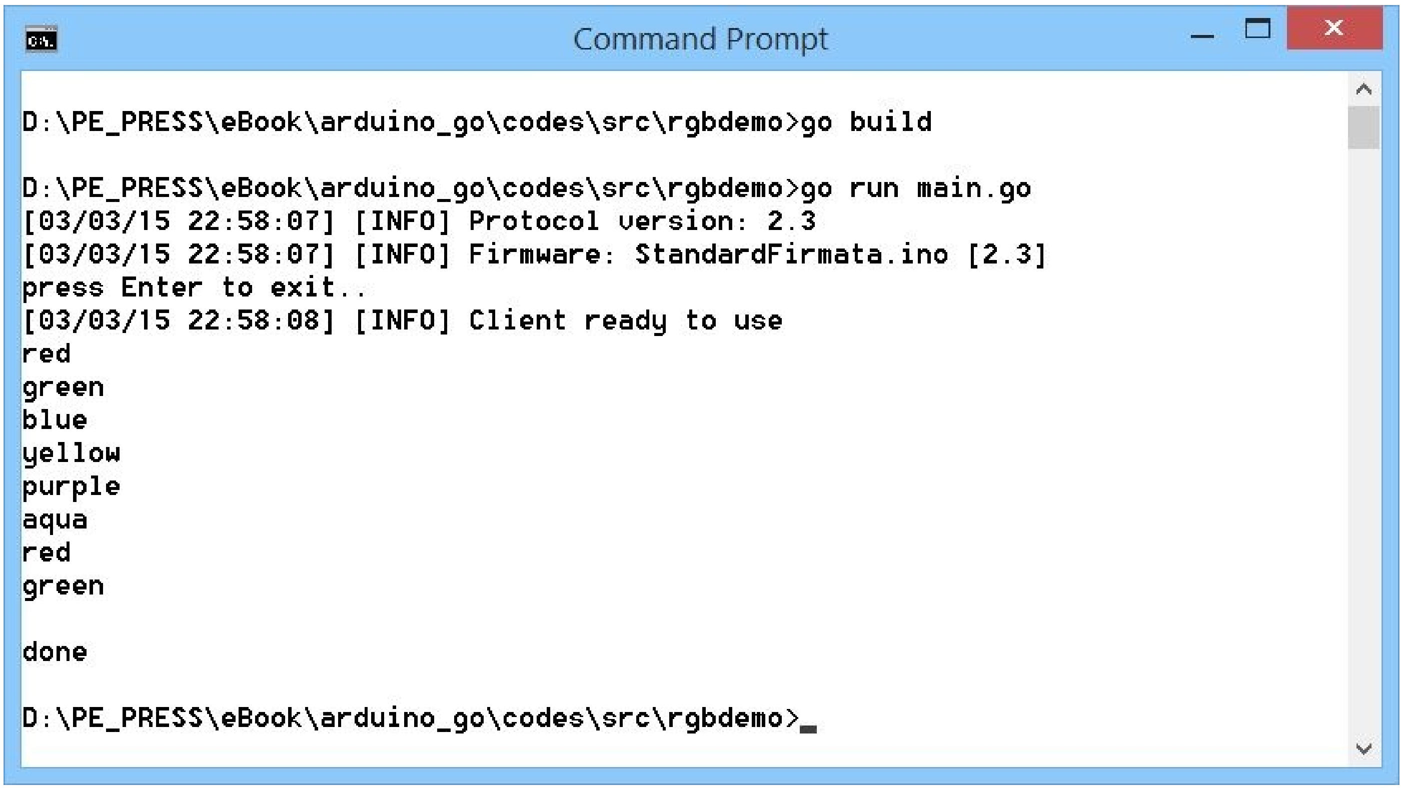
```
                SetColor(board,0, 255, 0)  // green
                fmt.Println("green")
                time.Sleep(2 * time.Second)
                SetColor(board,0, 0, 255)  // blue
                fmt.Println("blue")
                time.Sleep(2 * time.Second)
                SetColor(board,255, 255, 0)  // yellow
                fmt.Println("yellow")
                time.Sleep(2 * time.Second)
                SetColor(board,80, 0, 80)  // purple
                fmt.Println("purple")
                time.Sleep(2 * time.Second)
                SetColor(board,0, 255, 255)  // aqua
                fmt.Println("aqua")
                time.Sleep(2 * time.Second)

            }
        }()

        // press ENTER to exit
        fmt.Println("press Enter to exit..")
        var input string
        fmt.Scanln(&input)
        fmt.Println("done")
}

func SetColor(client *arduino.FirmataClient, red byte, green byte, blue

        client.AnalogWrite(1, red)
        client.AnalogWrite(3, green)
        client.AnalogWrite(9, blue)
}
```

Save this code and run it.

Now you can build and run this program. Don’t forget to connect your Arduino into PC via USB.

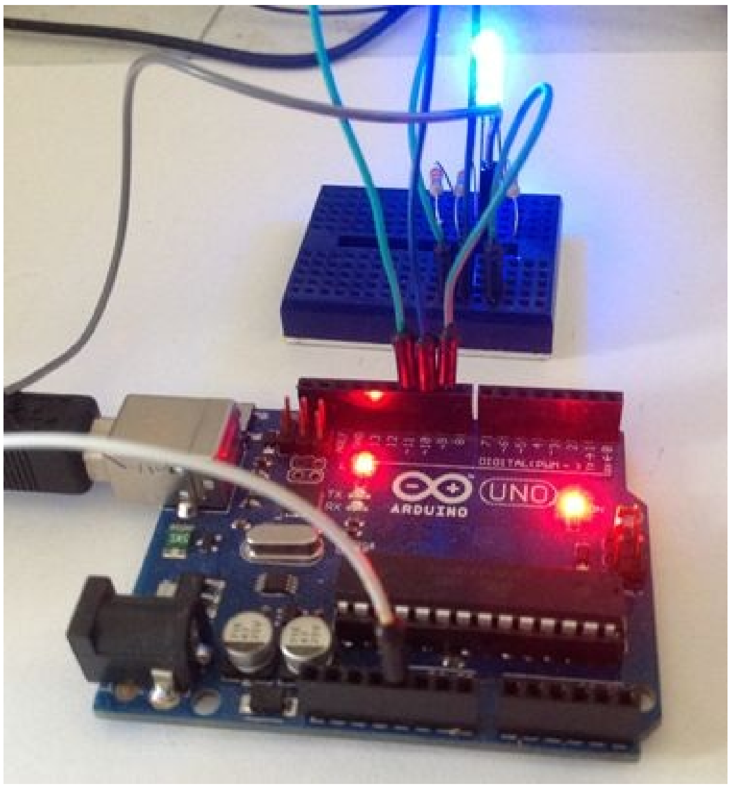
```
$ go build
$ go run main.go
```

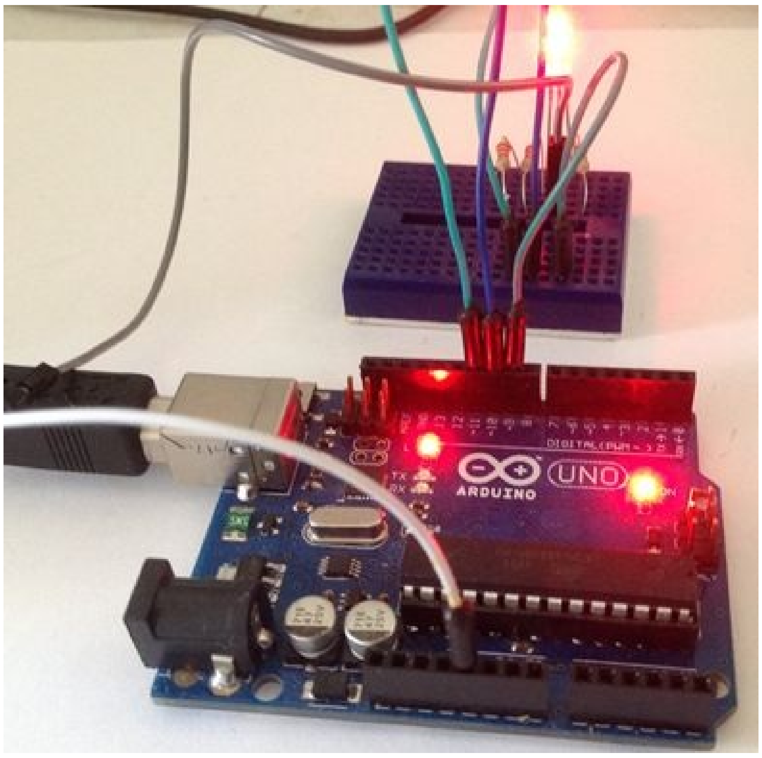
On console, you can see the output as below.



```
D:\PE_PRESS\eBook\arduino_go\codes\src\rgbdemo>go build

D:\PE_PRESS\eBook\arduino_go\codes\src\rgbdemo>go run main.go
[03/03/15 22:58:07] [INFO] Protocol version: 2.3
[03/03/15 22:58:07] [INFO] Firmware: StandardFirmata.ino [2.3]
press Enter to exit..
[03/03/15 22:58:08] [INFO] Client ready to use
red
green
blue
yellow
purple
aqua
red
green

done

D:\PE_PRESS\eBook\arduino_go\codes\src\rgbdemo>_
```

and you also see RGB LED blinking with different colors.




ARDUINO
UNO
DIGITAL (PWM ~)
ON
TX
RX
L

Source Code

You can download source code on
http://www.aguskurniawan.net/book/gocodes_123e1.zip .

Contact

If you have question related to this book, please contact me at aguskur@hotmail.com . My blog: http://blog.aguskurniawan.net