**4. PROGRAMMING ARDUINO USING SIMULINK SUPPORT PACKAGES**

**4.1 INTRODUCTION TO ARDUINO**

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.

Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP, MATLAB Simulink.) The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.

The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Net media’s BX-24, Phidgets, MIT's Handy board, and many others offer similar functionality. All of these tools take the messy details of microcontroller programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

* Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $15
* Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
* Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino
* Open source and extensible software- The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA, ATMEGA168 and ATMEGA2560 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

**4.2 BACK GROUND ABOUT ARDUINO**

In 2005, in [Ivrea, Italy](http://en.wikipedia.org/wiki/Ivrea,_Italy), a project was initiated to make a device for controlling student-built interaction design projects with less expense than with other prototyping systems available at the time. Founders Massimo Banzi and David Cuartielles named the project after [Arduino of Ivrea](http://en.wikipedia.org/wiki/Arduin_of_Ivrea), the main historical character of the town, and began producing boards in a small factory located in the same region as the computer company [Olivetti](http://en.wikipedia.org/wiki/Olivetti).

The Arduino project is a [fork](http://en.wikipedia.org/wiki/Fork_%28software_development%29) of the open source [Wiring](http://en.wikipedia.org/wiki/Wiring_%28development_platform%29) platform and is programmed using a Wiring-based language (syntax and libraries), similar to [C++](http://en.wikipedia.org/wiki/C%2B%2B) with some slight simplifications and modifications, and a [Processing](http://en.wikipedia.org/wiki/Processing_%28programming_language%29)-based [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE).

In October 2008, the Arduino Duemilanove was announced.

In March 2009, the Arduino Mega was announced.

As of May 2011[[update]](http://en.wikipedia.org/w/index.php?title=Arduino&action=edit), more than 400,000 Arduino units were in use around the world.

[](http://en.wikipedia.org/wiki/File:Arduino_001.jpeg)

Fig 4.1 First Arduino Board, late 2005

**4.3 ARDUINO SOFTWARE**

Software programs, called sketches, are created on a computer using the Arduino integrated development environment (IDE). The IDE enables you to write and edit code and convert this code into instructions that Arduino hardware understands. The IDE also transfers those instructions to the Arduino board (a process called uploading).

The Arduino [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is a [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) application written in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29), and is derived from the IDE for the [Processing programming language](http://en.wikipedia.org/wiki/Processing_%28programming_language%29) and the [Wiring](http://en.wikipedia.org/wiki/Wiring_%28development_platform%29) projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as [syntax highlighting](http://en.wikipedia.org/wiki/Syntax_highlighting), [brace matching](http://en.wikipedia.org/wiki/Brace_matching), and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit [make files](http://en.wikipedia.org/wiki/Makefiles) or run programs on a [command-line interface](http://en.wikipedia.org/wiki/Command-line_interface).

Arduino programs are written in [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) or [C++](http://en.wikipedia.org/wiki/C%2B%2B). The Arduino IDE comes with a [software library](http://en.wikipedia.org/wiki/Software_library) called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable [cyclic executive](http://en.wikipedia.org/wiki/Cyclic_executive) program:

* setup(): a function run once at the start of a program that can initialize settings
* loop(): a function called repeatedly until the board powers off

A typical first program for a microcontroller simply blinks an [LE](http://en.wikipedia.org/wiki/Light-emitting_diode)D on and off. In the Arduino environment, the user might write a program like this:

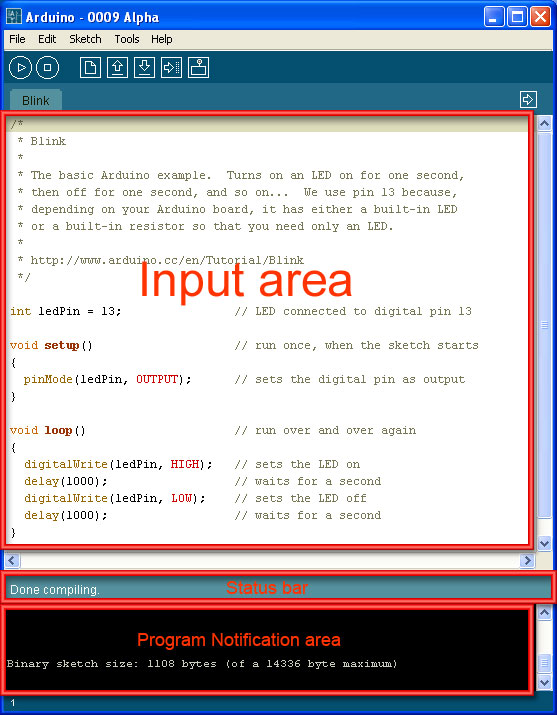


Figure 4.2 A screenshot of the Arduino IDE showing the "Blink" program, a simple beginner program.

define LED\_PIN 13

void setup () {

pinMode (LED\_PIN, OUTPUT); // enable pin 13 for digital output

}

void loop () {

digitalWrite (LED\_PIN, HIGH); // turn on the LED

delay (1000); // wait one second (1000 milliseconds)

digitalWrite (LED\_PIN, LOW); // turn off the LED

delay (1000); // wait one second

}

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground, a convenient feature for many simple tests. The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple [main() function](http://en.wikipedia.org/wiki/Main_function) at the bottom, to make it a valid C++ program. The Arduino IDE uses the [GNU tool chain](http://en.wikipedia.org/wiki/GNU_toolchain) and AVR Libc to compile programs, and uses avrdude to upload programs to the board. As the Arduino platform uses Atmel microcontrollers, Atmel's development environment, AVR Studio or the newer Atmel Studio, may also be used to develop software for the Arduino.

**4.4 ARDUINO HARDWARE**

The Arduino board is where the code you write is executed. The board can only control and respond to electricity, so specific components are attached to it to enable it to interact with the real world. These components can be sensors, which convert some and respond to electricity, so specific components are attached to it to enable it to aspect of the physical world to electricity so that the board can sense it, or actuators, which get electricity from the board and convert it into something that changes the world. Examples of sensors include switches, accelerometers, and ultrasound distance sensors. Actuators are things like lights and LEDs, speakers, motors, and displays.

There are a variety of official boards that you can use with Arduino software and a wide range of Arduino-compatible boards produced by members of the community. The most popular boards contain a USB connector that is used to provide power and connectivity for uploading your software onto the board. Figure 1-1 shows a basic board, the Arduino Uno.

You can get boards as small as a postage stamp, such as the Arduino Mini and Pro Mini; larger boards that have more connection options and more powerful processors, such as the Arduino Mega; and boards tailored for specific applications, such as the LilyPad for wearable applications, the Fio for wireless projects, and the Arduino Pro for embedded applications (standalone projects that are often battery-operated). Many other Arduino-compatible boards are also available, including the following:

* Arduino Nano, a tiny board with USB capability, from Gravitech.
* Bare Bones Board, a low-cost board available with or without USB capability, from Modern Device.
* Boarduino, a low-cost breadboard-compatible board, from Adafruit Industries.
* Seeeduino, a flexible variation of the standard USB board, from Seeed Studio Bazaar.
* Teensy and Teensy++, tiny but extremely versatile boards, from PJRC.

**4.5 SIMULINK SUPPORT PACKAGES FOR ARDUINO**

### To use the Simulink for programming Arduino we need have Simulink support packages for Arduino

### 4.5.1 INSTALL SUPPORT FOR ARDUINO HARDWARE

You can add support for Arduino® hardware to the Simulink® product. When you complete this process, and replace the firmware, you can run Simulink models on Arduino hardware.

The installation process adds the following items to your host computer:

* Third-party software development tools, such as the Arduino software with Mega 2560, Uno, Nano 3.0, Due, Leonardo, Mega ADK, Mini, Fio, Pro, Micro, Esplora, Robot Control Board, Robot Motor Board, Yun, and Lilypad USB board support.
* A Simulink block library for configuring and accessing Arduino sensors, actuators, and communication interfaces.
* Examples for getting started and learning about specific features.
* Documentation for Simulink Support Package for Arduino Hardware.

### 4.5.2 INSTALL, UPDATE, OR UNINSTALL SUPPORT PACKAGE

#### 4.5.2.1 INSTALL SUPPORT PACKAGE

1. On the MATLABHome tab, in the Environment section, click Add-Ons > Get Hardware Support Packages.
2. In the Add-On Manager window, find and click the support package, and then click Install.

#### 4.5.2.2 UPDATE SUPPORT PACKAGE

On the MATLAB Home tab, in the Environment section, click Add-Ons > Check for Updates > Hardware Support Packages.

#### 4.5.2.3 UNINSTALL SUPPORT PACKAGE

1. On the MATLAB Home tab, in the Environment section, click Add-Ons > Manage Add-Ons.
2. In the Add-On Manager window, find and click the support package, and then click Uninstall.

### 4.5.2.4 COMPLETE ADDITIONAL SETUP TASKS

1. If you clicked Setup Later at the end of the Add-On Manager installation process, you can restart hardware setup process.
   * On the MATLAB Home tab, in the Environment section, click Add-Ons > Manage Add-Ons. When the Add-On Manager opens, click Setup
   * Enter [targetupdater](http://in.mathworks.com/help/matlab/ref/targetupdater.html) in the MATLAB Command Window.
2. Follow the instructions and default settings provided by Support Package Installer to complete the installation. For more information about the options on a particular screen, click the Help button.

### 4.6 GETTING STARTED WITH ARDUINO HARDWARE

This example shows how to use Simulink Support Package for Arduino Hardware to run a Simulink® model on Arduino board.

**Supported Hardware**

* Arduino Esplora
* Arduino Fio
* Arduino Leonardo
* Arduino Lilypad USB
* Arduino Mega 2560
* Arduino Mega ADK
* Arduino Micro
* Arduino Mini
* Arduino Nano 3.0
* Arduino Pro
* Arduino Uno
* Arduino Due

The provided model is pre-configured for Arduino Mega 2560 and can be run on any of the board listed in the "Supported Hardware" section, by changing the "Hardware board" parameter in the configuration parameters dialog box of the model as described in Task 4 of this example.

### 4.6.1 INTRODUCTION

Simulink Support Package for Arduino Hardware enables you to create and run Simulink models on Arduino board. The target includes a library of Simulink blocks for configuring and accessing Arduino sensors, actuators and communication interfaces. Additionally, the target enables you to monitor and tune algorithms running on Arduino board from the same Simulink models from which you developed the algorithms.

In this example you will learn how to create and run a simple Simulink model on Arduino board. See other examples for Arduino board to learn how to use External mode and to learn how to implement more complex algorithms.

### 4.6.2 REQUIRED HARDWARE

To run this example you will need the following hardware:

* Supported Arduino board
* USB cable
* LED
* 220 Ohm resistor
* Breadboard wires
* Small breadboard (recommended)

### Model

The following figure shows the example model:

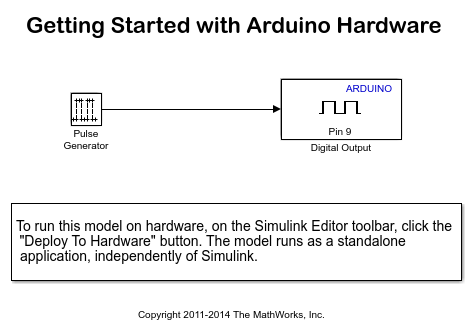


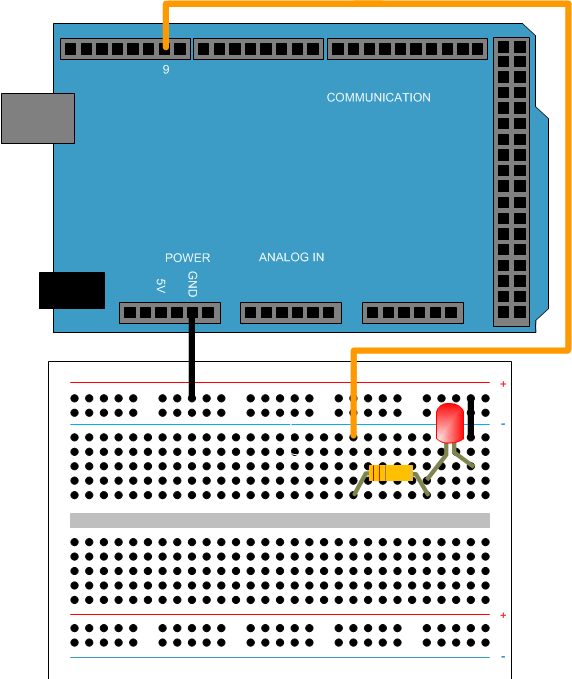
Figure 4.3 Blink LED using Simulink

### Task 1 - Connect an LED to an Arduino Output Pin

In this task, you will connect an LED to an Arduino output pin so you can see changes in the logical state of the pin.

1. Attach one end of the 220 Ohm resistor to output pin 9 on the Arduino board. Use the recommended breadboard and the breadboard wires.

2. Attach the long leg (positive) of the LED to the resistor. Attach the short leg (negative) to the ground pin on the Arduino board.



### Figure 4.4 Connecting LED to Arduino

### Task 2 - Review Arduino Block Library

Simulink Support Package for Arduino Hardware provides an easy way to create algorithms that use Arduino sensors and actuators by using the blocks that can be added to your Simulink model. The blocks are used to configure the associated sensors and actuators, as well as to read and write data to them.

1. Enter slLibraryBrowser at the MATLAB® prompt. This opens the Simulink Library Browser.

2. In the Simulink Library Browser, navigate to Simulink Support Package for Arduino Hardware > Common.

3. Double-click the Digital Output block. Review the block mask, which contains a description of the block and parameters for configuring the associated Arduino digital output pin.

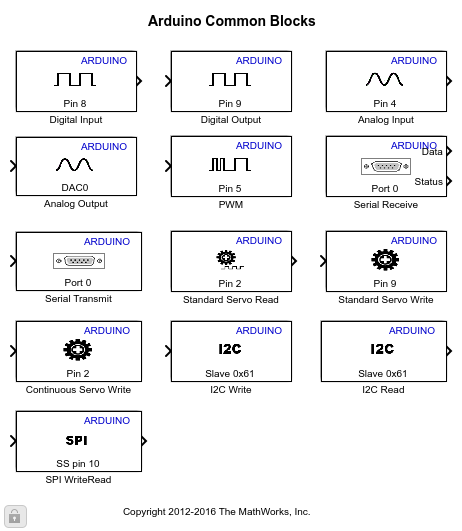


Figure 4.5 Simulink blocks for Arduino

### Task 3 - Create a Model for Arduino Hardware

In this task, you will create a simple Simulink model that changes the state of the Arduino digital output pin.

1. In MATLAB, select HOME > New > Simulink Model.

2. Drag the Pulse Generator block from the Simulink Sources library to your model.

3. Double-click the Pulse Generator block. Set the Pulse type to parameter to Sample based and set the Sample time parameter to 0.1 second.

4. Drag the Digital Output block to the model. Use the default block settings.

5. Connect the Pulse Generator block to the Digital Output block.

### Task 4 - Configure and Run the Model on Supported Arduino Hardware

In this task, you will configure and run your model on the supported Arduino board.

1. Connect the Arduino board to your computer with a USB cable.

2**.** In your Simulink model, click Simulation > Model Configuration Parameters to open Configuration Parameters dialog.

3. Select the Hardware Implementation pane and select your required Arduino hardware from the Hardware board parameter list. Do not change any other settings.

4. Click OK.

5. In your Simulink model, click the Deploy to Hardware button on the toolbar. The model will now be deployed to the connected Arduino hardware.

6**.** Look at the LED attached to pin 9. The LED should blink one time every second.

7**.** Save your model.

### Summary

This example introduced the workflow for creating an algorithm from a Simulink model and then running it on the supported Arduino board. In this example we can see that:

* Simulink Support Package for Arduino Hardware provides blocks for configuring, reading from and writing to Arduino sensors and actuators.
* We can use the Deploy to Hardware button to configure and run the model on supported Arduino board.

### 4.7 PWM

Generate square waveform on specified analog output pin

**Library block:**

Simulink Support Package for Arduino® Hardware/Common



**4.7.1 DESCRIPTION**

Use pulse-width modulation (PWM) to change the duty-cycle of square-wave pulses output by a PWM pin on the Arduino hardware. PWM enables a digital output to provide a range of different power levels, similar to that of an analog output.

The value sent to the block input determines the width of the square wave, called duty-cycle, that the target hardware outputs on the specified PWM pin. The range of valid input values is 0 to 255.

For example:

* Sending the maximum value, 255, to the block input produces 100% duty-cycle, which results in full power on a PWM pin.
* Sending the minimum value, 0, to the block input produces 0% duty-cycle, which results in no power on a PWM pin.
* Sending an intermediate value to the block input produces a proportional duty-cycle and power output on a PWM pin. For example, sending 204 to the block input produces 80% duty cycle and power (204/255 = 0.8).
* Sending out-of-range values, such as 500 or -500, to the block input has the same effect as sending the maximum or minimum input values.

The frequency of the PWM signal on most pins is ~490 Hz. On the Arduino Uno and similar boards, pins 5 and 6 have a frequency of ~980 Hz.

The block input inherits the data type of the upstream block, and internally converts it to uint8.

Some limitations:

* With Arduino Uno, Nano, Pro, Fio, Mini hardware, the Arduino PWM block cannot use digital pins 9 or 10 when the model contains Servo blocks.
* With Arduino Mega 2560, Mega ADK hardware, the Arduino PWM block cannot use digital pins 11 or 12 when the model contains more than 12 Servo blocks.
* With Arduino Due hardware, the Arduino PWM block cannot use digital pins 9 or 10 when the model contains Servo blocks.

### 4.7.2 PARAMETERS

**Pin number**

Enter the number of the PWM pin.

Do not assign the same pin number for different types of blocks as this may cause resource management conflicts.