Introduction to ARDUINO

An Open-Source Prototyping Platform

Hans-Petter Halvorsen
Contents

1. Overview
2. Installation
3. Arduino Starter Kit
4. Arduino TinkerKit
5. Arduino Examples
6. LabVIEW Interface for Arduino Toolkit

http://home.hit.no/~hansha/?equipment=arduino
Arduino is a single-board microcontroller to make using electronics in multidisciplinary projects more accessible.

The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM.

The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.
Hardware Specifications

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328)
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz
1. **POWER In (Barrel Jack)** - Can be used with either a 9V or 12V wall-wart or battery.

2. **POWER In (USB Port)** - Provides power and communicates with your board when plugged into your computer via USB.

3. **LED (RX: Receiving)** - This shows when the Arduino is receiving data (such as when being programmed).

4. **LED (TX: Transmitting)** - This shows when your Arduino is transmitting data (such as when running a program).

5. **LED (Pin 13: Troubleshooting)** - This LED is incorporated into your sketch to show if your program is running properly.

6. **PINS (ARef, Ground, Digital, Rx, Tx)** - These various pins can be used for inputs, outputs, power, and ground. // See Diagram Below

7. **LED (Indicates Arduino is ON)** - This is a simple power indicator LED.

8. **Reset Button** - This is a way to manually reset your Arduino, which makes your code restart.

9. **ICSP PINS (Upload Code without Bootloader)** - This is for “In-Circuit Serial Programming,” used if you want to bypass the bootloader.

10. **PINS (Analog In, Power In, Ground, Power Out, Reset)** - These various pins can be used for inputs, outputs, power, and ground. // See Diagram Below
The header pins are one of the most important parts for putting our example circuits together. Take a moment and locate the input/output ports of your Arduino Uno.
Alternatives

Alt 1 (no wiring)

Alt 2 (wiring)

Alt 3 (soldering)

TinkerKit
Sensor Shield

Breadboard

Arduino UNO

Arduino Proto Shield
Introduction to Arduino by Hans-Petter Halvorsen
A breadboard is used to create circuits and connect different sensors and actuators to the Arduino board.
Software

Arduino IDE/Sketch

```cpp
// This example code is in the public domain.

// include the TinkerKit library
#include <TinkerKit.h>

// creating the object 'led' that belongs to the 'TKled' class.
// and giving the value to the desired output pin

TKled led(0);  // set the LED on

void setup() {  
    // nothing here
}

void loop() {
    led.on();  // set the LED on
    delay(1000);  // wait for a second
    led.off();  // set the LED off
    delay(1000);  // wait for a second
}

// done uploading.

Binary sketch size: 1 992 bytes (of a 32 256 byte maximum)
Binary sketch size: 1 992 bytes (of a 32 256 byte maximum)
```
A typical program written in Sketch:

```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

This simple program makes a LED connected to pin 13 blink.
Installation

http://arduino.cc/en/Main/Software
Arduino Starter Kit

Arduino.cc/starterkit
Kit Contents

- 1 Arduino Projects Book (170 pages)
- 1 Arduino UNO board rev.3
- 1 USB cable
- 1 Breadboard
- 1 Easy-to-assemble wooden base
- 19v battery snap
- 70 Solid core jumper wires
- 2 Stranded jumper wires
- 6 Photoresistor
- 3 Potentiometer 10 kilohm
- 10 Pushbuttons
- 1 Temperature sensor
- 1 Tilt sensor
- 1 LCD alphanumeric (16x2 characters)
- 1 LED (bright white)
- 1 LED (RGB)
- 8 LEDs (red)
- 8 LEDs (green)
- 8 LEDs (yellow)
- 3 LEDs (blue)

- 1 Small DC motor 6/9V
- 1 Small servo motor
- 1 Piezo capsule
- 1 H-bridge motor driver
- 2 Optocouplers
- 5 Transistor
- 2 Mosfet transistors
- 5 Capacitors 100nF
- 3 Capacitors 100uF
- 5 100pF capacitor
- 5 Diodes
- 3 Transparent gels (red, green, blue)
- 1 Male pins strip (40x1)
- 20 Resistors 220 ohm
- 5 Resistors 560 ohm
- 5 Resistors 1 kilohm
- 5 Resistors 4.7 kilohm
- 10 Resistors 10 kilohm
- 5 Resistors 1 megohm
- 5 Resistors 10 megohm
Starter Kit

Contents

• Arduino Uno - the latest Arduino USB board, fully assembled and tested.

• 6' USB A to B cable - USB provides power for up to 500mA (enough for most projects) and is ample length to connect to your desktop or laptop USB port.

• Mini breadboard - Excellent for making circuits and connections off the Arduino. Breadboard may come in various colors.

• Male to Male jumper wires - These are high quality wires that allow you to connect the female headers on the Arduino to the components and breadboard.

• Flex Sensor - Originally designed for the Nintendo Power Glove, now you too can measure flex!

• SoftPot - Measure position along the softpot by looking at the change in resistance. It's like a touch sensitive volume slider.

• Photocell - A sensor to detect ambient light. Perfect for detecting when a drawer is opened or when night-time approaches.

• Thermistor - A sensor for detecting ambient temperature and temperature changes.

• Tri-Color LED - Because everyone loves a blinky. Use this LED to PWM mix any color you need.

• Basic LEDs - Light emitting diodes make great general indicators.

• Linear trim pot - Also known as a variable resistor, this is a device commonly used to control volume, contrast, and makes a great general user control input.

• Buzzer - Make wonderful, brain splitting noises, alarms, and possibly music!

• 12mm button - Because big buttons are easier to hit.

• 330 Ohm Resistors - 5 current limiting resistors for LEDs, and strong pull-up resistors.

• 10k Ohm Resistors - These make excellent pull-ups, pull-downs, and current limiters.
SparkFun Inventors Kit for Arduino (SIK)

sparkfun.com/sikcode
SIK Kit Contents

• Arduino Uno R3

• Arduino and Breadboard Holder

• New and Improved SIK Manual

• New and Improved SIK Carrying Case

• Translucent Red Bread Board

• 74HC595 Shift Register

• 2N2222 Transistors

• 1N4148 Diodes

• DC Motor with wires

• Small Servo

• 5V Relay

• TMP36 Temp Sensor

• Flex sensor

• Softpot

• 6' USB Cable

• Jumper Wires

• Photocell

• Tri-color LED

• Red and Yellow LEDs

• 10K Trimpot

• Piezo Buzzer

• Big 12mm Buttons

• 330 and 10K Resistors
Ohms Law

\[ U = RI \]

Resistor

Diode

Transistor
Resistors

Serial:

$R_{eq} = R_1 + R_2 + \ldots + R_n$

Parallel:

\[
\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}
\]

If we have 2 in parallel:

\[
R_{eq} = R_1 || R_2 = \frac{R_1 R_2}{R_1 + R_2}
\]
To distinguish left from right there is a gap between the C and D bands.

- band A is first significant figure of component value (left side)
- band B is the second significant figure (Some precision resistors have a third significant figure, and thus five bands.)
- band C is the decimal multiplier
- band D if present, indicates tolerance of value in percent (no band means 20%)

<table>
<thead>
<tr>
<th>Color</th>
<th>Significant figures</th>
<th>Multiplier</th>
<th>Tolerance</th>
<th>Temp. Coefficient (ppm/K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>0</td>
<td>$\times 10^0$</td>
<td>–</td>
<td>250</td>
</tr>
<tr>
<td>Brown</td>
<td>1</td>
<td>$\times 10^1$</td>
<td>±1%</td>
<td>100</td>
</tr>
<tr>
<td>Red</td>
<td>2</td>
<td>$\times 10^2$</td>
<td>±2%</td>
<td>50</td>
</tr>
<tr>
<td>Orange</td>
<td>3</td>
<td>$\times 10^3$</td>
<td>–</td>
<td>15</td>
</tr>
<tr>
<td>Yellow</td>
<td>4</td>
<td>$\times 10^4$</td>
<td>(±5%)</td>
<td>25</td>
</tr>
<tr>
<td>Green</td>
<td>5</td>
<td>$\times 10^5$</td>
<td>±0.5%</td>
<td>20</td>
</tr>
<tr>
<td>Blue</td>
<td>6</td>
<td>$\times 10^6$</td>
<td>±0.25%</td>
<td>10</td>
</tr>
<tr>
<td>Violet</td>
<td>7</td>
<td>$\times 10^7$</td>
<td>±0.1%</td>
<td>5</td>
</tr>
<tr>
<td>Gray</td>
<td>8</td>
<td>$\times 10^8$</td>
<td>±0.05% (±10%)</td>
<td>1</td>
</tr>
<tr>
<td>White</td>
<td>9</td>
<td>$\times 10^9$</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Gold</td>
<td>–</td>
<td>$\times 10^{-1}$</td>
<td>±5%</td>
<td>J</td>
</tr>
<tr>
<td>Silver</td>
<td>–</td>
<td>$\times 10^{-2}$</td>
<td>±10%</td>
<td>K</td>
</tr>
<tr>
<td>None</td>
<td>–</td>
<td>–</td>
<td>±20%</td>
<td>M</td>
</tr>
</tbody>
</table>

For example, a resistor with bands of yellow, violet, red, and gold will have first digit 4 (yellow), second digit 7 (violet), followed by 2 (red) zeros: 4,700 ohms. Gold signifies that the tolerance is ±5%, so the real resistance could lie anywhere between 4,465 and 4,935 ohms.
Arduino TinkerKit

No Wiring and breadboard is needed, just “plug and play”

The TinkerKit Shield is mounted on top of the Arduino board
TinkerKit Overview

Sensors and Actuators

Wires

TinkerKit Shield
TinkerKit Sensors

- Accelerometer
- Button
- Gyroscope 1x
- Gyroscope 4x
- Hall
- Joystick
- Light Sensor
- Linear Pot
- Rotary Pot
- Thermistor
- Tilt
- Touch
TinkerKit
Actuators

High Power LED  LED blue 10mm  LED blue 5mm  LED green 10mm
LED green 5mm  LED red 10mm  LED red 5mm  LED yellow 10mm
LED yellow 5mm  Micro Servo  Mosfet  Relay
Installation

http://www.tinkerkit.com
Examples

1. LED
2. Push Button
3. Temperature
4. Potentiometer
// include the TinkerKit library
#include <TinkerKit.h>

// creating the object 'led' that belongs to the 'TKLed' class
TKLed led(00);

void setup() {
    //nothing here
}

void loop() {
    led.on(); // set the LED on
    delay(1000); // wait for a second
    led.off(); // set the LED off
    delay(1000); // wait for a second
}
This module is a **SENSOR**. The connector is an **OUTPUT** which must be connected to one of the **INPUT** connectors on the TinkerKit Shield.
The Thermistor is a resistor whose resistance varies significantly (more than in standard resistors) with temperature. **Output:** This module’s output approaches 5V as the temperature increases. As the temperature decreases, it approaches 0V. When connected to an input on the Arduino using the TinkerKit Shield, expect to read values between 0 and 1023

```cpp
#include <TinkerKit.h>

TKThermistor therm(I0);

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  C = therm.readCelsius();
  F = therm.readFahrenheit();

  Serial.print("Analog reading: ");
  Serial.print(therm.read());
  Serial.print("tC: ");
  Serial.println(C);
  Serial.print("tF: ");
  Serial.println(F);

  delay(1000);
}
```
Potentiometer

A Potentiometer is a commonly used variable resistor. Turning the knob, you vary the output voltage between 0 and 5V. This value is sent through the middle pin of the pot.

Output: This module outputs 5V when turned in one direction, and 0v when turned in the opposite way. When connected to an input on the Arduino using the TinkerKit Shield, you can expect to read values between 0 and 1023.

```cpp
#include <TinkerKit.h>

// creating the object 'pot' that belongs to the 'TKPotentiometer' class
TKPotentiometer pot(I0);

// creating the object 'led' that belongs to the 'TKLed' class
TKLed led(O0);

int brightnessVal = 0;  // value read from the pot

void setup() {
  // initialize serial communications at 9600 bps
  Serial.begin(9600);
}

void loop() {
  // read the potentiometer's value:
  brightnessVal = pot.read();

  // set the led brightness
  led.brightness(brightnessVal);

  // print the results to the serial monitor:
  Serial.print("brightness = ");
  Serial.println(brightnessVal);

  // wait 10 milliseconds before the next loop
  delay(10);
}
```
Exercises

1. Thermostat:
   - Turn a led on when the temperature is below a value.
   - Read the threshold value from a potentiometer
Arduino Examples

1. LED
2. Push Button
3. Temperature
4. Potentiometer
5. PWM
6. Servo Motor
7. DC Motor
8. Relay
int led = 13;

void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);  // turn the LED on (HIGH is the voltage level)
    delay(1000);               // wait for a second
    digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);               // wait for a second
}
LED cont.

- LED: Make sure the short leg, marked with flat side, goes into the negative position (-).
- 330Ω Resistor: The color banding should read orange-orange-brown-gold. The component legs can go in either hole.
- Jumper Wire: All jumper wires work the same. They are used to connect two points in the circuit. Different colored combinations of colors is completely acceptable.

Introduction to Arduino by Hans-Petter Halvorsen
Push Button

Pushbuttons or switches connect two points in a circuit when you press them. This example turns on the built-in LED on pin 13 when you press the button.

```c
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin = 13;       // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
    pinMode(buttonPin, INPUT);
}

void loop(){
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);

    // check if the pushbutton is pressed.
    // if it is, the buttonState is HIGH:
    if (buttonState == HIGH) {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }
    else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
}
```

Introduction to Arduino by Hans-Petter Halvorsen
const int temperaturePin = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    float voltage, degreesC, degreesF;
    // First we'll measure the voltage at the analog pin. Normally
    // we'd use analogRead(), which returns a number from 0 to 1023.
    // Here we've written a function (further down) called
    // getVoltage() that returns the true voltage (0 to 5 Volts)
    // present on an analog input pin.
    voltage = getVoltage(temperaturePin);
    // Now we'll convert the voltage to degrees Celsius.
    // This formula comes from the temperature sensor datasheet:
    degreesC = (voltage - 0.5) * 100.0;
    // While we're at it, let's convert degrees Celsius to Fahrenheit.
    // This is the classic C to F conversion formula:
    degreesF = degreesC * (9.0/5.0) + 32.0;
    // Now we'll use the serial port to print these values
    // to the serial monitor!
    Serial.print("voltage: ");
    Serial.print(voltage);
    Serial.print("  deg C: ");
    Serial.print(degreesC);
    Serial.print("  deg F: ");
    Serial.println(degreesF);
    delay(1000); // repeat once per second (change as you wish!)
}

float getVoltage(int pin)
{
    return (analogRead(pin) * 0.004882814);
    // This equation converts the 0 to 1023 value that analogRead()
    // returns, into a 0.0 to 5.0 value that is the true voltage
    // being read at that pin.
}
Temperature cont.
int sensorPin = 0;    // The potentiometer is connected to analog pin 0
int ledPin = 13;      // The LED is connected to digital pin 13

void setup() // this function runs once when the sketch starts up
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    int sensorValue;
    sensorValue = analogRead(sensorPin);

    digitalWrite(ledPin, HIGH);       // Turn the LED on
    delay(sensorValue);              // Pause for
    digitalWrite(ledPin, LOW);       // Turn the LED off
    delay(sensorValue);              // Pause for
}

Measure the position of a potentiometer and use it to control the blink rate of an LED.

Turn the knob to make it blink faster or slower!
Potentiometer cont.
PWM

PWM - Pulse Width Modulation

The shocking truth behind analogWrite():

We’ve seen that the Arduino can read analog voltages (voltages between 0 and 5 Volts) using the `analogRead()` function. Is there a way for the Arduino to output analog voltages as well?

The answer is no... and yes. The Arduino does not have a true analog voltage output. But, because the Arduino is so fast, it can fake it using something called **PWM** ("Pulse-Width Modulation").

The Arduino is so fast that it can blink a pin on and off almost 1000 times per second. PWM goes one step further by varying the amount of time that the blinking pin spends HIGH vs. the time it spends LOW. If it spends most of its time HIGH, a LED connected to that pin will appear bright. If it spends most of its time LOW, the LED will look dim. Because the pin is blinking much faster than your eye can detect, the Arduino creates the illusion of a "true" analog output.
Servo Motor

#include <Servo.h>  // servo library

Servo servol;  // servo control object

void setup()
{
    servol.attach(9);
}

void loop()
{
    int position;

    // To control a servo, you give it the angle you'd like it
    // to turn to. Servos cannot turn a full 360 degrees, but you
    // can tell it to move anywhere between 0 and 180 degrees.

    servol.write(90);    // Tell servo to go to 90 degrees
    delay(1000);         // Pause to get it time to move

    servol.write(180);   // Tell servo to go to 180 degrees
    delay(1000);         // Pause to get it time to move

    servol.write(0);     // Tell servo to go to 0 degrees
    delay(1000);         // Pause to get it time to move
}

By varying the pulse of voltage a servo receives, you can move a servo to a specific position. For example, a pulse of 1.5 milliseconds will move the servo 90 degrees. In this circuit, you’ll learn how to use PWM (pulse width modulation) to control and rotate a servo.
Servo Motor cont.
DC Motor

In order to make a DC motor work with Arduino we need to use a transistor, which can switch a larger amount of current than the Arduino can handle.

const int motorPin = 9;

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

void loop()
{
    motorOnThenOff();
}

void motorOnThenOff()
{
    int onTime = 3000; // milliseconds to turn the motor on
    int offTime = 3000; // milliseconds to turn the motor off

    digitalWrite(motorPin, HIGH); // turn the motor on (full speed)
    delay(onTime); // delay for onTime milliseconds
    digitalWrite(motorPin, LOW); // turn the motor off
    delay(offTime); // delay for offTime milliseconds
}
DC Motor cont.
Here we'll use pulse-width modulation (PWM) to vary the speed of a motor.

```c
const int motorPin = 9;

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

void loop()
{
  motorOnThenOffWithSpeed();
}

// This function alternates between two speeds.
// Try different values to affect the timing and speed.
void motorOnThenOffWithSpeed()
{
  int Speed1 = 200;  // between 0 (stopped) and 255 (full speed)
  int Time1 = 3000;  // milliseconds for speed 1

  int Speed2 = 50;   // between 0 (stopped) and 255 (full speed)
  int Time2 = 3000;  // milliseconds to turn the motor off

  analogWrite(motorPin, Speed1);  // turns the motor On
  delay(Time1);                   // delay for onTime milliseconds
  analogWrite(motorPin, Speed2);  // turns the motor Off
  delay(Time2);                   // delay for offTime milliseconds
}
Relay
1. Thermostat:
   - Turn a led on when the temperature is below a value.
   - Read the threshold value from a potentiometer.
The LabVIEW Interface for Arduino (LIFA) Toolkit is a FREE download that allows developers to acquire data from the Arduino microcontroller and process it in the LabVIEW Graphical Programming environment.

https://decibel.ni.com/content/groups/labview-interface-for-arduino
Installation

https://decibel.ni.com/content/groups/labview-interface-for-arduino
Examples

1. LED
2. Push Button
3. Temperature
4. Potentiometer
Exercises

1. Thermostat:
   - Turn a led on when the temperature is below a value.
   - Read the threshold value from a potentiometer
References

- Arduino Web Site: http://arduino.cc
- TinkerKit Web Site: http://www.tinkerkit.com
- LabVIEW Interface for Arduino: https://decibel.ni.com/content/groups/labview-interface-for-arduino
- Arduino: http://home.hit.no/~hansha/?equipment=arduino
Introduction to

Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway
www.usn.no

E-mail: hans.p.halvorsen@hit.no
Blog: http://home.hit.no/~hansha/