ARDUINO IDE

Contents

Download the Arduino IDE

The open-source Arduino environment makes it easy to write code and upload it to the i/o board (Windows).

http://downloads.arduino.cc/arduino-1.0.6-windows.exe

SUPPORT AND OTHER DOWNLOADS

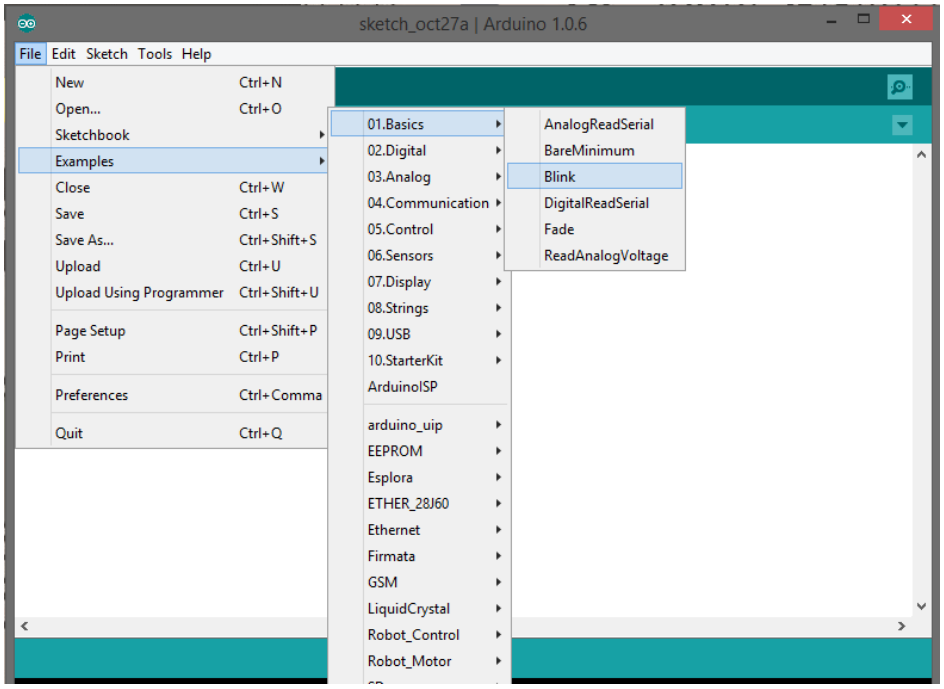
http://arduino.cc/en/Main/Software

Installing drivers for the Arduino Uno with Windows

- Plug in your board and wait for Windows to begin it's driver installation process. After a few moments, the process will fail, despite its best efforts
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".
- Right click on the "Arduino UNO (COmxx)" port and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named "arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named "Arduino UNO.inf"
- Windows will finish up the driver installation from there

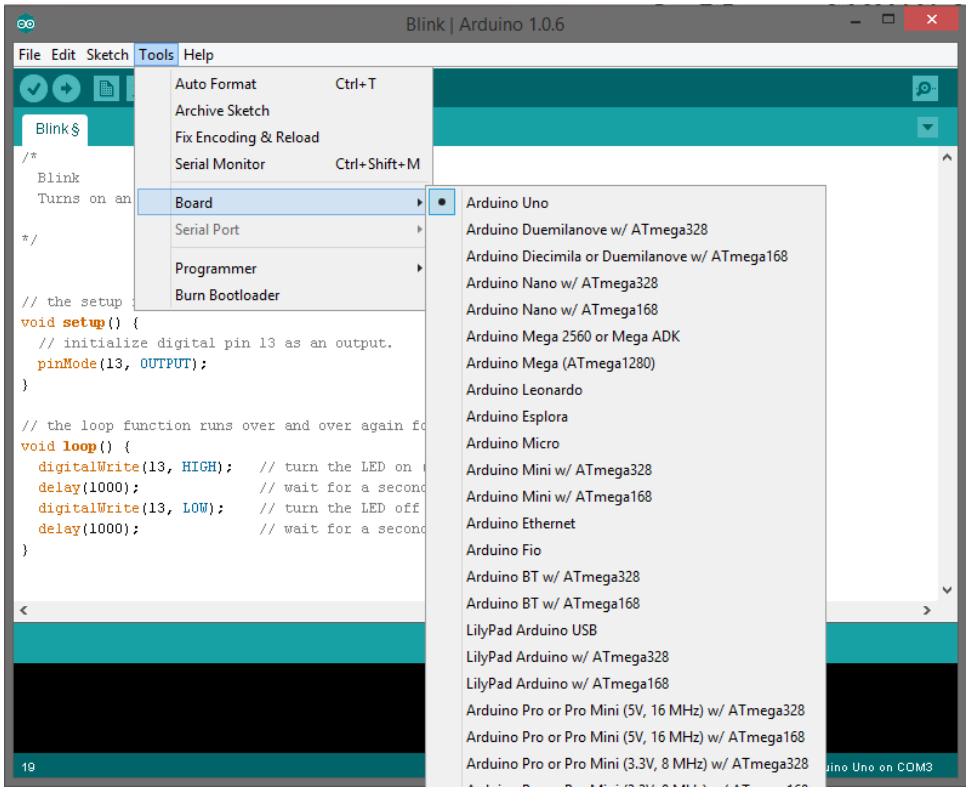
Launch the Arduino application

Open the LED blink example sketch: File > Examples > 1.Basics > Blink.

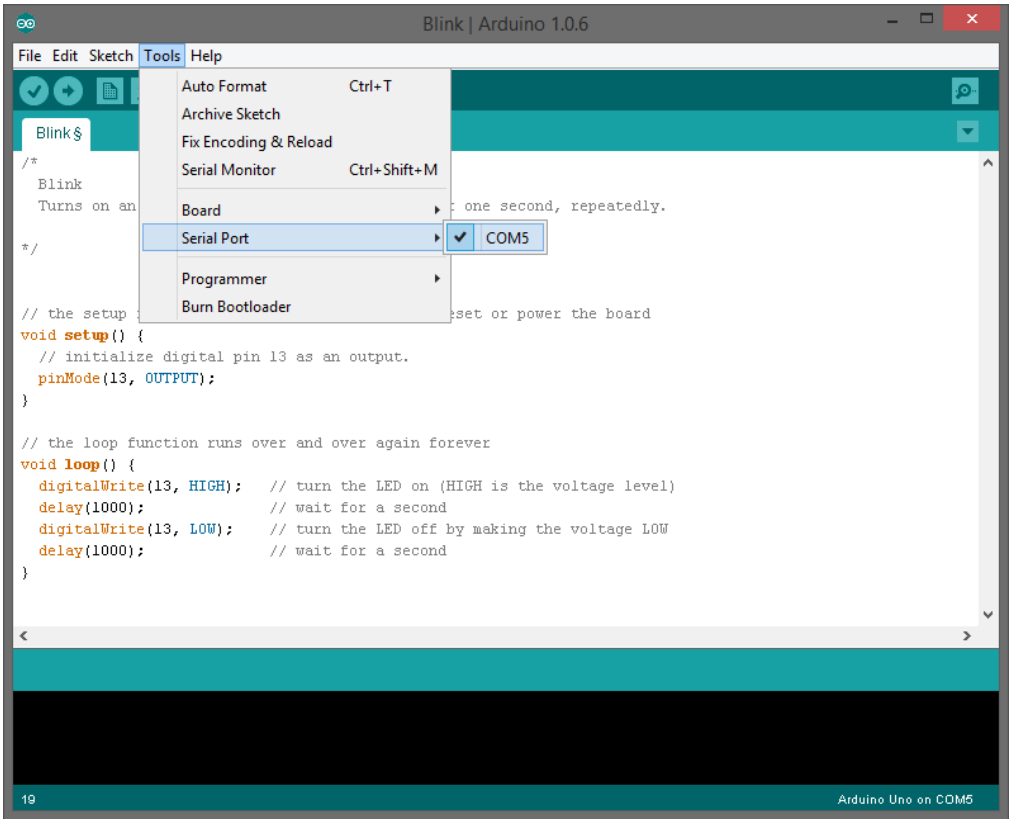


ARDUINO ENVIRONMENT

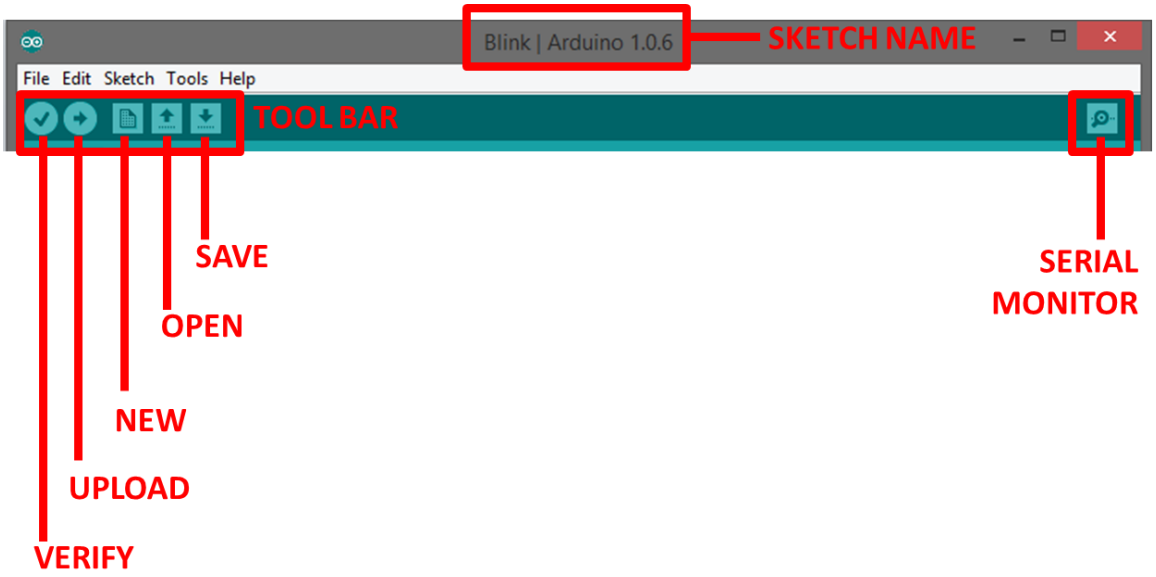
SELECTING BOARD TYPE



SELECTING SERIAL PORT / COM PORT



THE ENVIRONMENT



PARTS OF THE SKETCH

COMMENTS

- Comments can be anywhere
- Comments created with // or /* and */
- Comments do not affect code
- You may not need comments, but think about the community!

SETUP

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

The setup function comes before the loop function and is necessary for all Arduino sketches

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

The setup header will never change, everything else that occurs in setup happens inside the curly brackets

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

Outputs are declare in setup, this is done by using the pinMode function
This particular example declares digital pin # 13 as an output, remember to use CAPS

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  Serial.begin(9600);
}
```

Serial communication also begins in setup
This particular example declares Serial communication at a baud rate of 9600. More on Serial later.

```
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
  digitalWrite(12, HIGH);
}
```

You can also create internal pullup resistors in setup, to do so digitalWrite the pin HIGH
This takes the place of the pullup resistors currently on your circuit 7 buttons

Setup, Interrupts

void setup ( ) {
attachInterrupt (interrupt, function, mode) }

You can designate an interrupt function to Arduino pins # 2 and 3
This is a way around the linear processing of Arduino

Interrupt: the number of the interrupt, 0 or 1, corresponding to Arduino pins # 2 and 3 respectively
Function: the function to call when the interrupt occurs
Mode: defines when the interrupt should be triggered

- ***LOW*** whenever pin state is low
- ***CHANGE*** whenever pin changes value
- ***RISING*** whenever pin goes from low to high
- ***FALLING*** whenever pin goes from low to high

Don't forget to CAPITALIZE

OPERATORS

- The equals sign
  = is used to assign a value
  == is used to compare values
- And & Or
- && is “and”
- || is “or”

Variables

Basic variable types:

- Boolean
- Integer
- Character

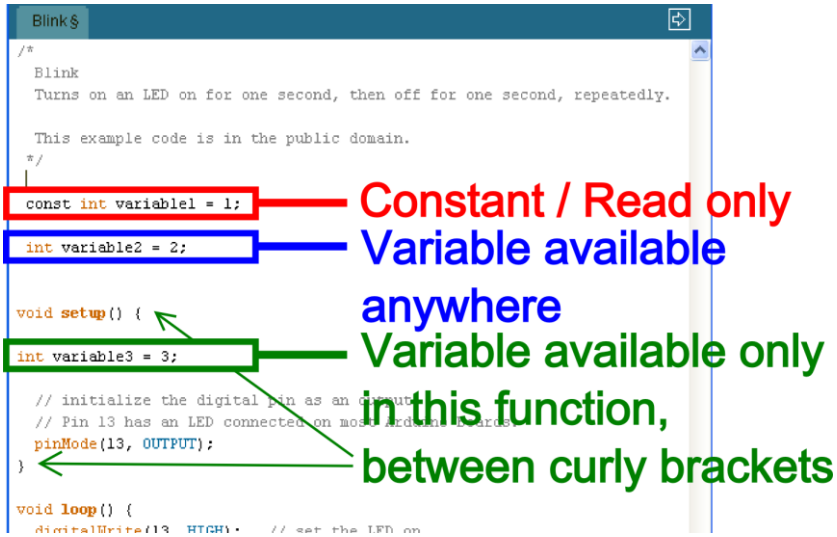
Declaring varialbles:

Boolean: boolean variableName;
Integer: int variableName;
Character: char variableName;
String: stringName [ ];

Assigning variables:

Boolean: variableName = true;
or variableName = false;
Integer: variableName = 32767;
or variableName = -32768;
Character: variableName = ‘A’;
or stringName = “rdl”;

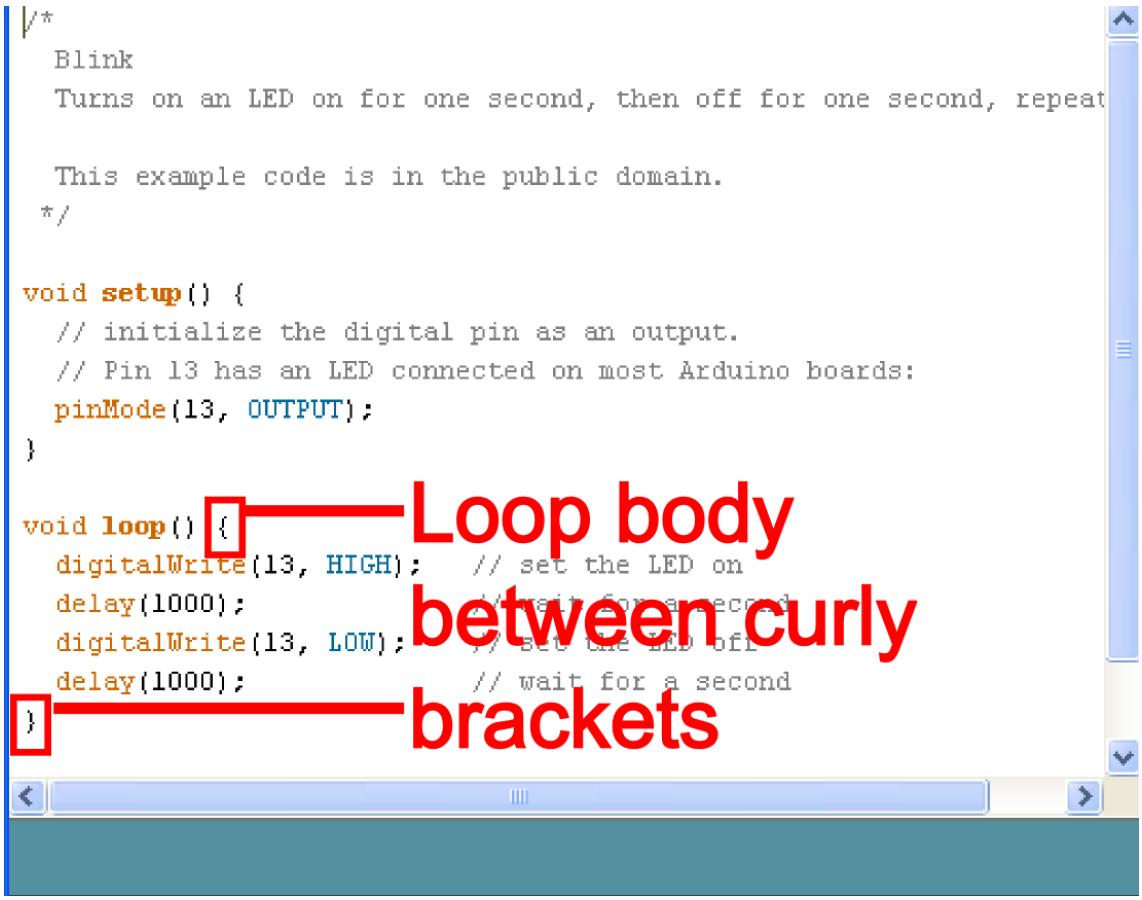
Variable scope



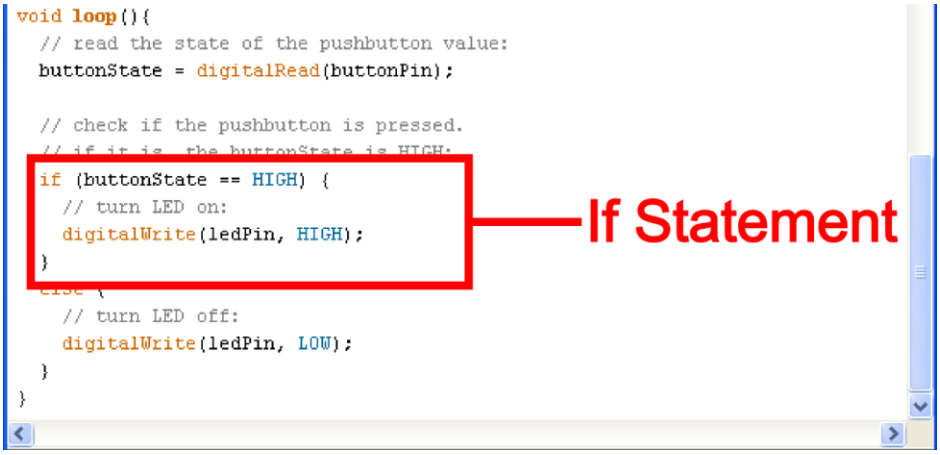
LOOP

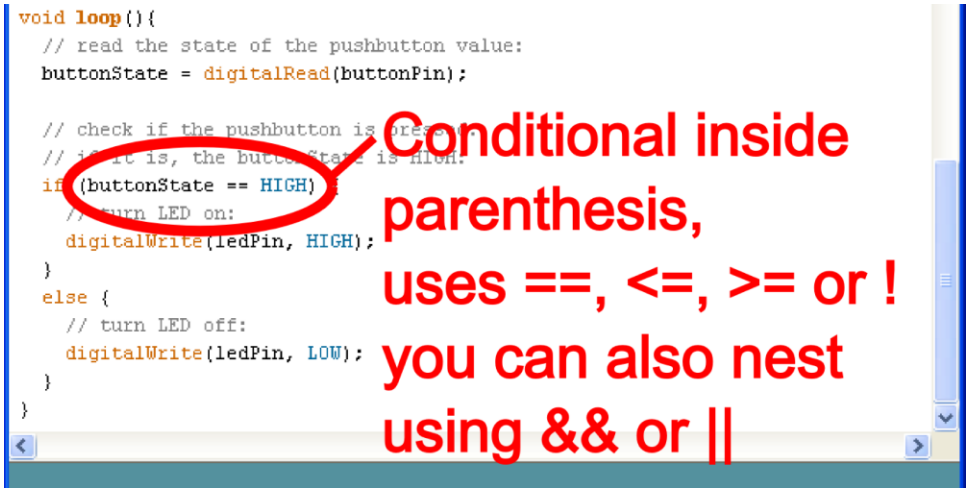
Basic Repetition

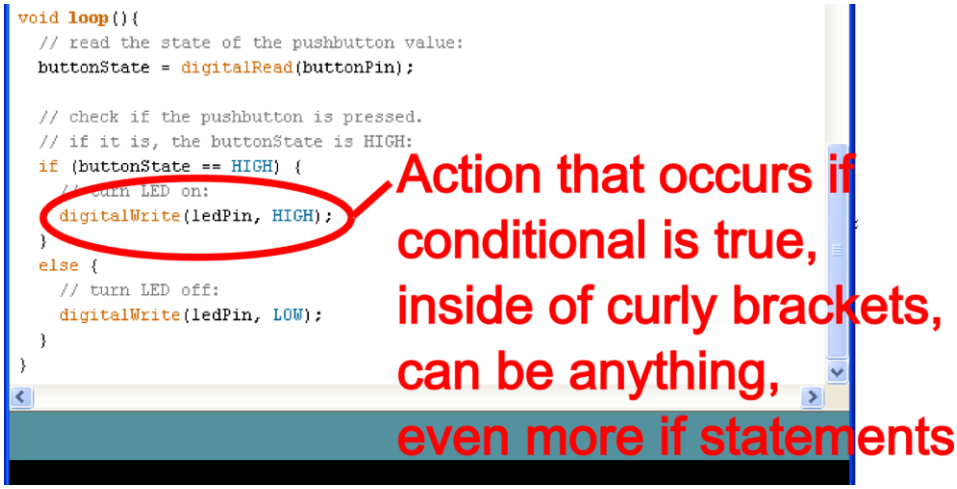
- loop
- For
- while



Inside void loop (){}








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);
}
}
Else, optional

EXAMPLE SKETCHES

Go to “File > Examples” in Arduino IDE

AnalogReadSerial

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

DigitalReadSerial

```
/*
  DigitalReadSerial
 Reads a digital input on pin 2, prints the result to the serial monitor

 This example code is in the public domain.
 */

// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);        // delay in between reads for stability
}
```

Serial Event

```
/*
 Serial Event example

 When new serial data arrives, this sketch adds it to a String.
 When a newline is received, the loop prints the string and
 clears it.

 A good test for this is to try it with a GPS receiver
 that sends out NMEA 0183 sentences.

 Created 9 May 2011
 by Tom Igoe

 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/SerialEvent

 */

String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

void setup() {
  // initialize serial:
  Serial.begin(9600);
  // reserve 200 bytes for the inputString:
  inputString.reserve(200);
}

void loop() {
  // print the string when a newline arrives:
  if (stringComplete) {
    Serial.println(inputString);
    // clear the string:
    inputString = "";
    stringComplete = false;
  }
}

/*
  SerialEvent occurs whenever a new data comes in the
 hardware serial RX.  This routine is run between each
 time loop() runs, so using delay inside loop can delay
 response.  Multiple bytes of data may be available.
 */
void serialEvent() {
```

```
  while (Serial.available()) {
    // get the new byte:
    char inChar = (char)Serial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```