

TM

ARDUINO

MICROCONTROLLERS – Revision

- A microcontroller can be considered as a very small and simple version of a computer on a single IC which is used for a specific purpose.
- It has a CPU, flash memory ,RAM, EEPROM and many on- chip peripherals .

MAIN TERMS USED IN MCU

- Clock : Oscillating Signal with fixed time period
- Clock source : Generally Crystal Oscillators
- Flash memory : Stores the program
- SRAM : Volatile Memory
- EEPROM : Permanent Memory
- VCC : Power Supply
- GND : Ground

MAIN TERMS USED IN MCU

- PORT : The point where data internal to the MCU comes out. Simply, a combination of pins in an MCU is a PORT. A port contains 8 GPIO(General Purpose Input Output pins).
- ADC : ADC ports receive analog signals and convert them into digital number within certain numerical range depending on the resolution. ADC channels are shared with PORT A of AVR.
- Motor Driver : It is an external driver circuit which acts as a bridge between IC and motors.

Development Boards

- With the microcontrollers done, lets move on to the next step.
- Microcontrollers give the signals to the actuators. How do we order them to give the signals?
- It is done by the use of development boards.

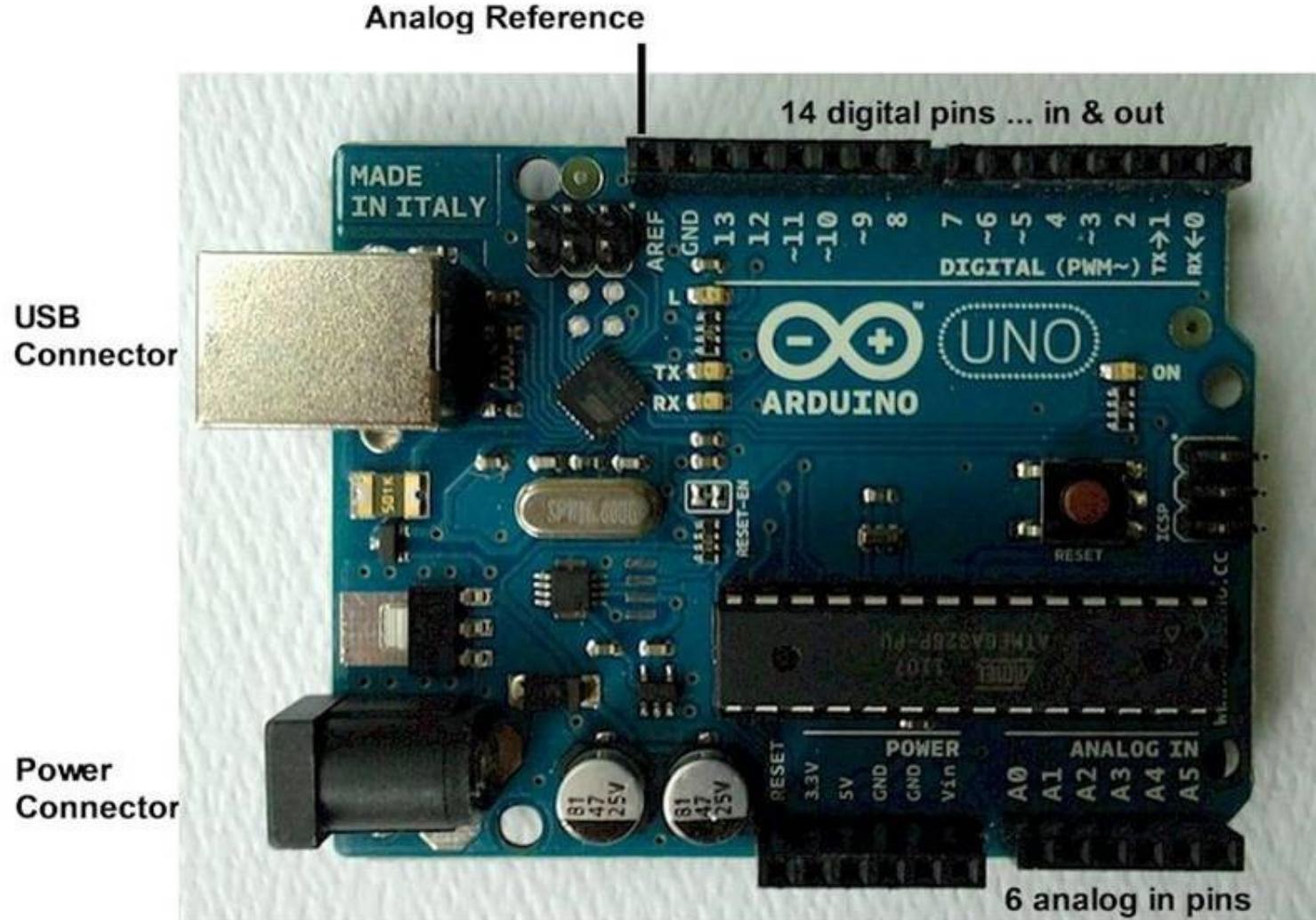
Development Boards

- Printed circuit boards that provide all the circuitry necessary for a useful control task like I/O circuit, clock generator, stored program etc.
- Here we are going to discuss about 2 types of development boards in particular.
 - 1. Arduino
 - 2. AVR

ARDUINO

- Arduino is an AVR-based prototyping board with an emphasis on ease of use.
- It has separate pins for digital and analog purposes.
- There are 6 *analog in pins* and 14 *digital in/out pins* ground, Vcc ...
- It consists of PWM output pins separately.
- It also features a USB interface allowing serial communication through a USB device, eliminating the need for a separate AVR programmer.

ARDUINO



Why ARDUINO ?

- The main reason is its programming environment.
- It includes many helpful libraries .
- While programming Arduino, you are least bothered with the internal hardware and registers of the microcontroller. All you do is call the functions written in the libraries (which are already provided)
- Generally used if you want to prototype your project very fast, and are not much concerned about the programming part.

AVR

- AVR is a microcontroller manufactured by Atmel
- We use atmega16A microcontroller.
- Atmega 16A is used in AVR and atmega 328 is used in arduino.
- In this, communication between PC and MCU is done using a programmer.
- It consists of a on-board driver.

ATmega16A

(XCK/T0)	PB0	1	40	PA0 (ADC0)
(T1)	PB1	2	39	PA1 (ADC1)
(INT2/AIN0)	PB2	3	38	PA2 (ADC2)
(OC0/AIN1)	PB3	4	37	PA3 (ADC3)
(SS)	PB4	5	36	PA4 (ADC4)
(MOSI)	PB5	6	35	PA5 (ADC5)
(MISO)	PB6	7	34	PA6 (ADC6)
(SCK)	PB7	8	33	PA7 (ADC7)
RESET		9	32	AREF
VCC		10	31	GND
GND		11	30	AVCC
XTAL2		12	29	PC7 (TOSC2)
XTAL1		13	28	PC6 (TOSC1)
(RXD)	PD0	14	27	PC5 (TDI)
(TXD)	PD1	15	26	PC4 (TDO)
(INT0)	PD2	16	25	PC3 (TMS)
(INT1)	PD3	17	24	PC2 (TCK)
(OC1B)	PD4	18	23	PC1 (SDA)
(OC1A)	PD5	19	22	PC0 (SCL)
(ICP1)	PD6	20	21	PD7 (OC2)

Comparison between Arduino & AVR

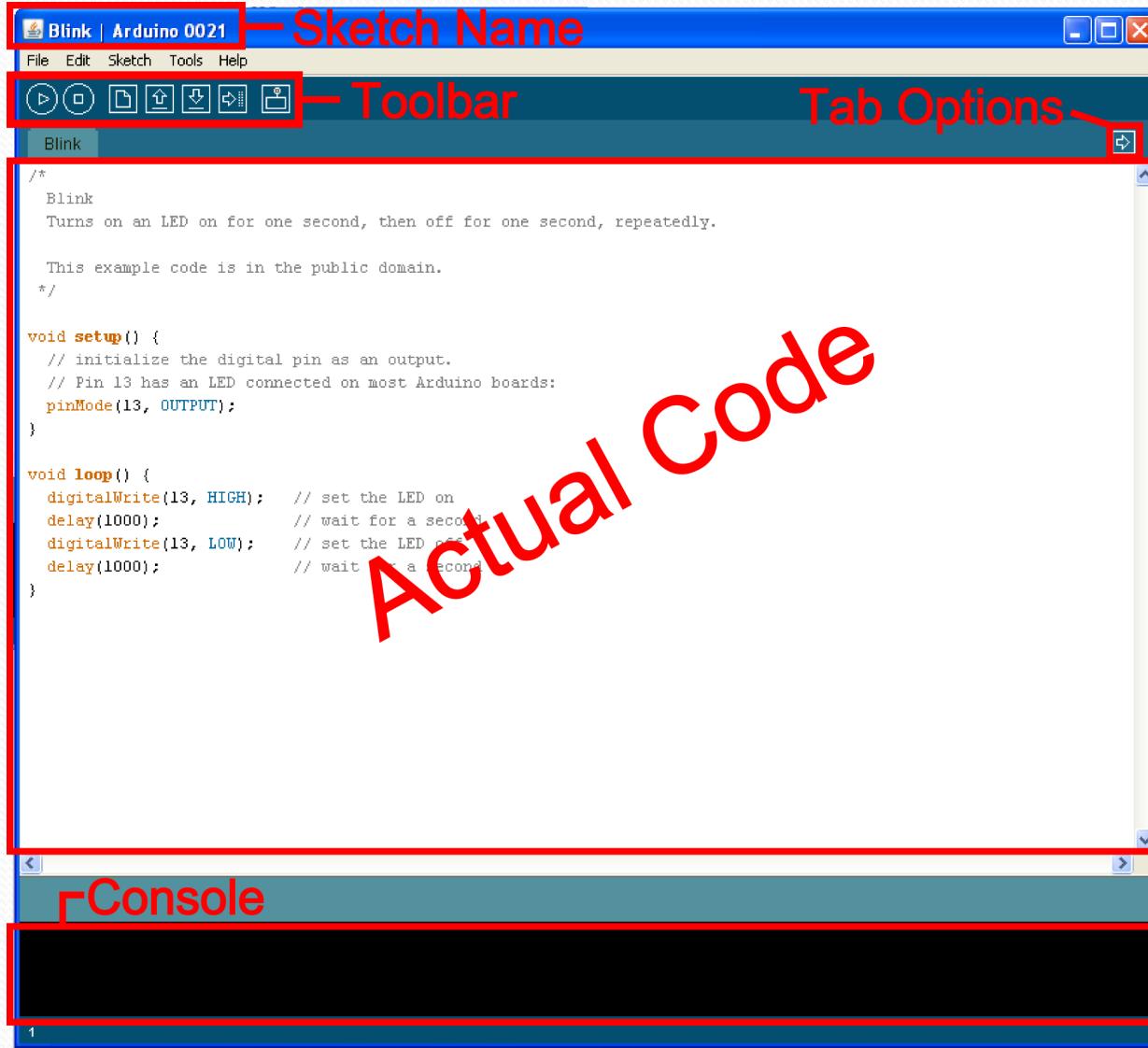
Arduino

- It is easier to use
- It requires little knowledge
- No need to know the inbuild circuits
- It is not as powerful as AVR
- Meant for an amateur

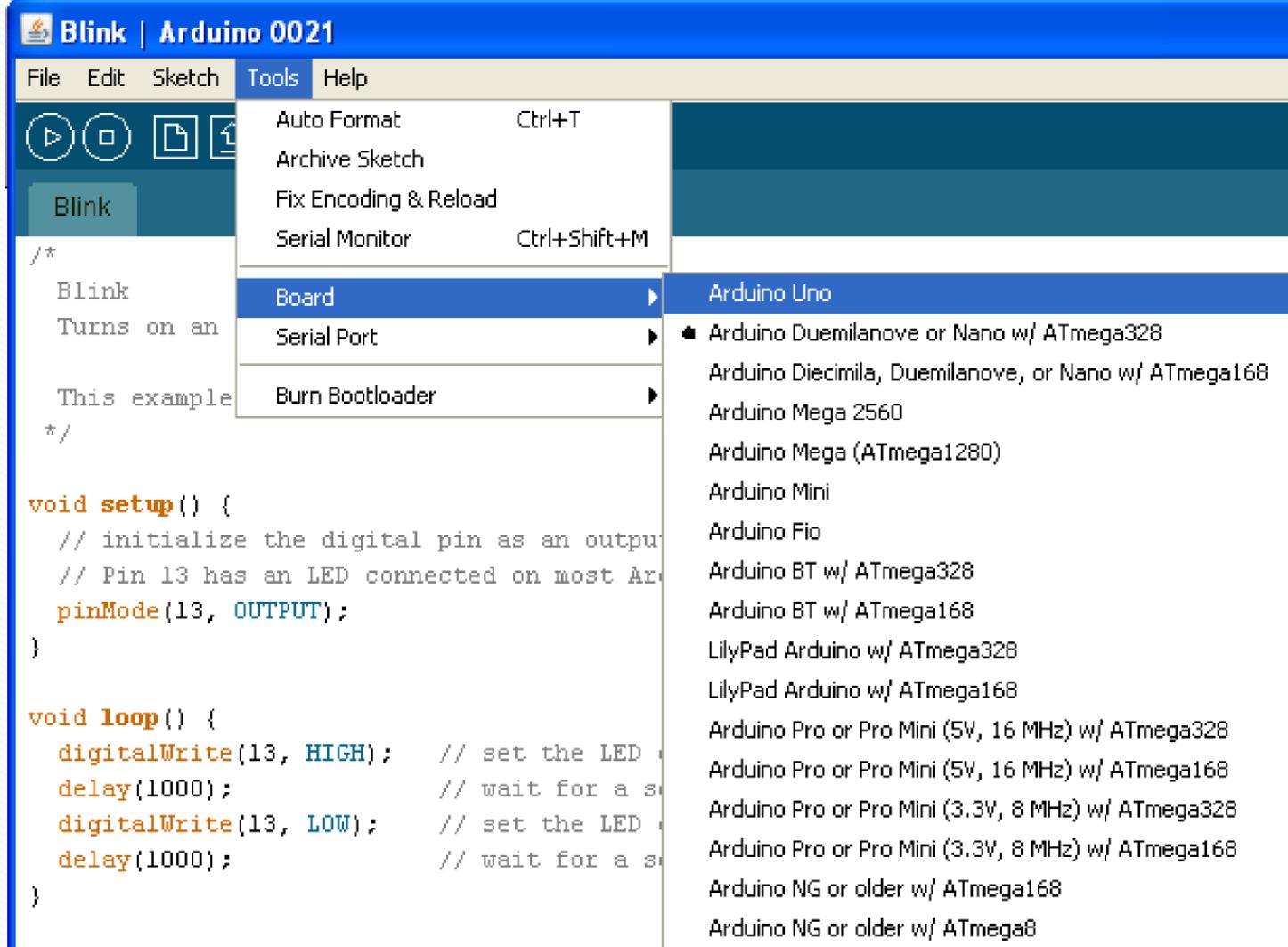
AVR

- It is more difficult to use
- It requires much knowledge
- One needs to know about how the internal circuits are implemented
- More powerful than Arduino
- Meant for a

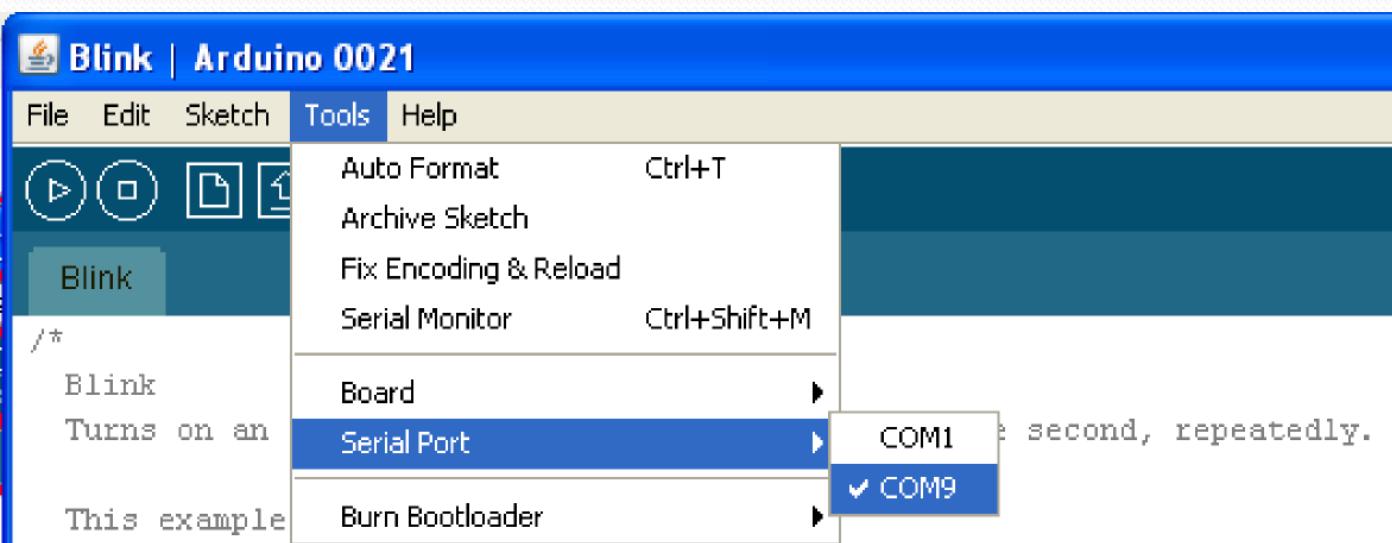
The Arduino Environment



Board Type



Serial Port / COM Port



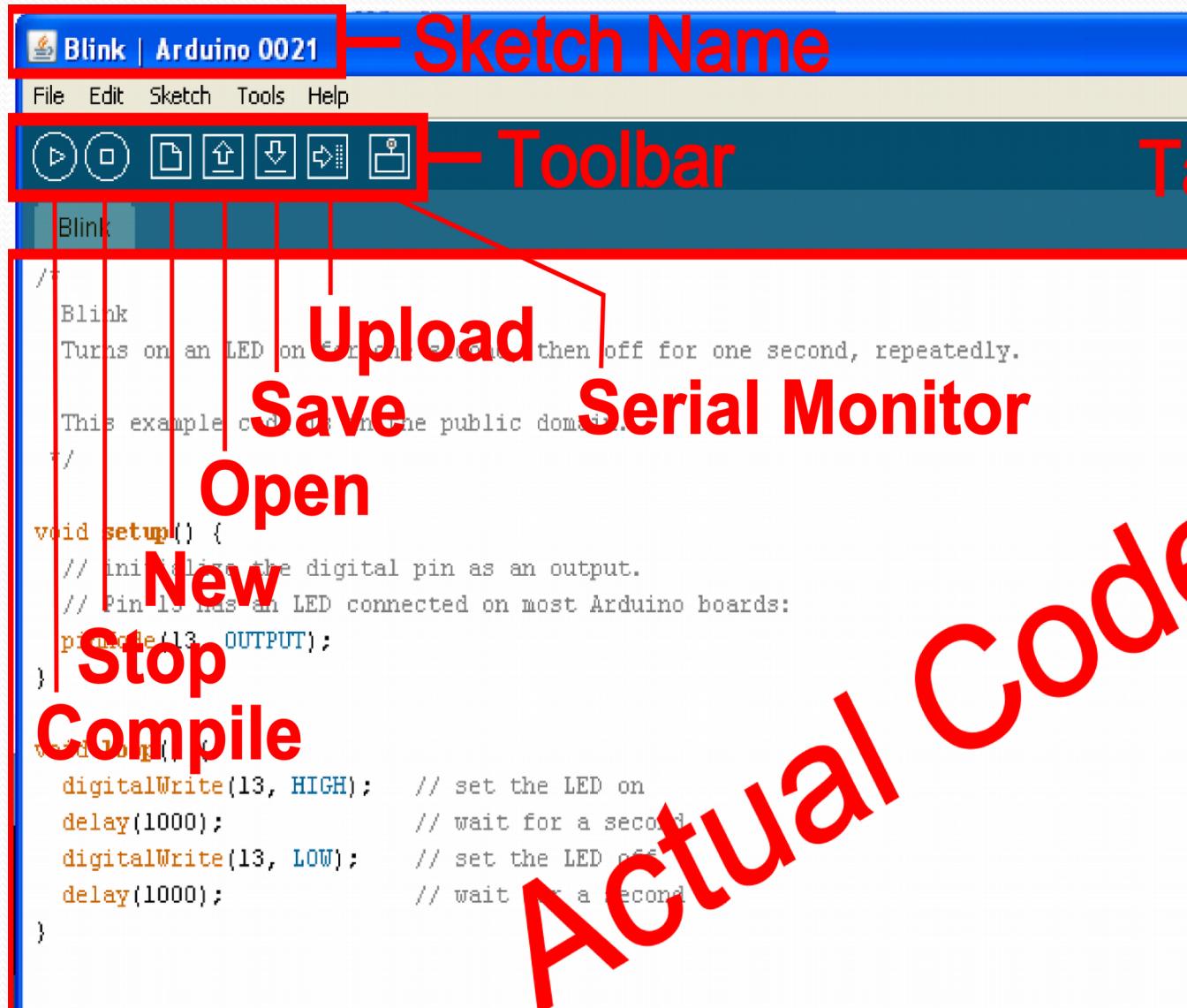
The screenshot shows the Arduino IDE interface. The title bar reads "Blink | Arduino 0021". The menu bar includes File, Edit, Sketch, Tools, and Help. The Tools menu is open, showing options: Auto Format (Ctrl+T), Archive Sketch, Fix Encoding & Reload, Serial Monitor (Ctrl+Shift+M), Board, Serial Port, and Burn Bootloader. The "Serial Port" submenu is expanded, showing options: COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, and COM9. The option "COM1" is highlighted with a blue selection bar. The main code editor displays the "Blink" sketch, which blinks an LED connected to digital pin 13.

```
/*
Blink
Turns on an LED
This example code is in the public domain.

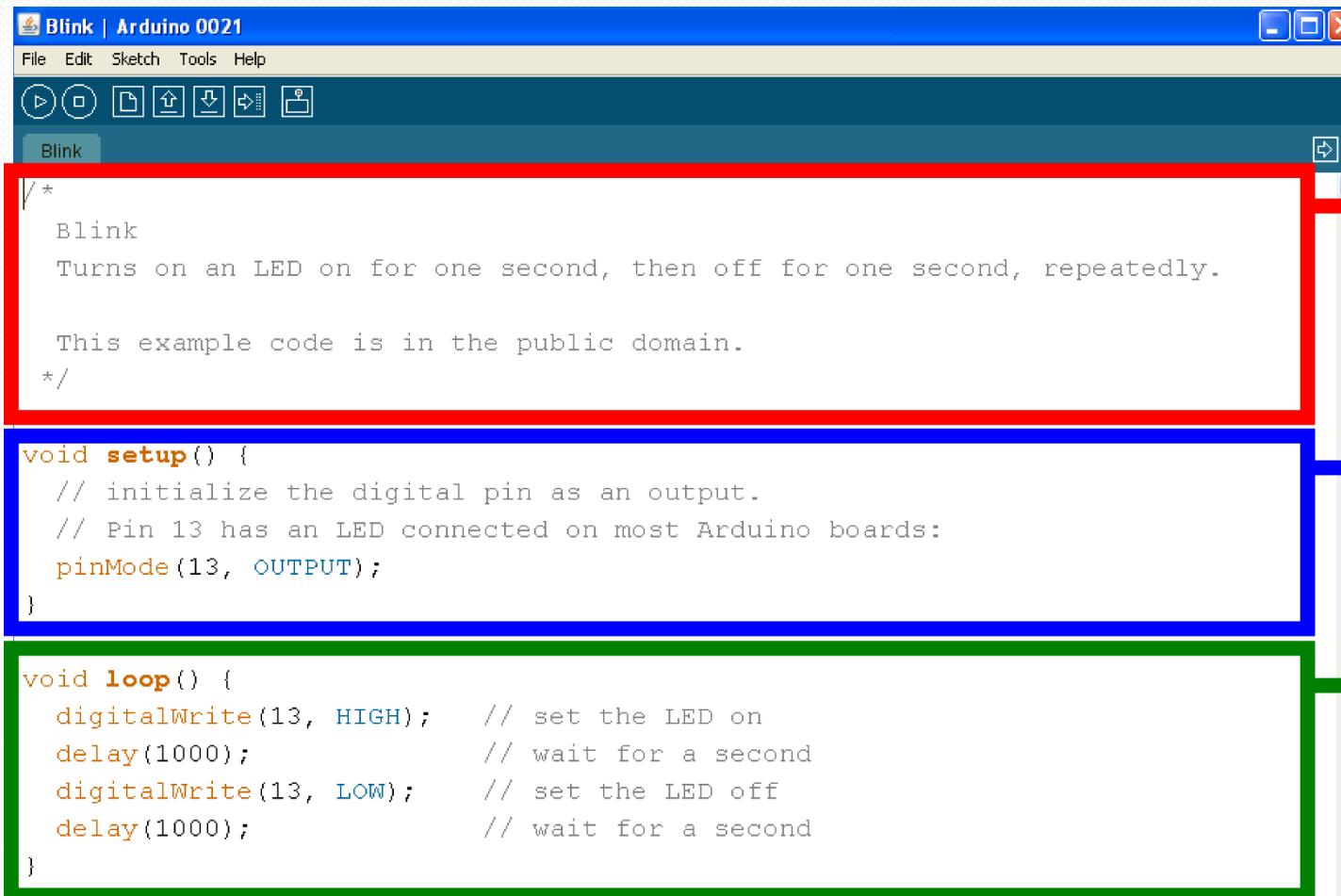
void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);      // set the LED on
  delay(1000);                // wait for a second
  digitalWrite(13, LOW);       // set the LED off
  delay(1000);                // wait for a second
}
```

The Environment



Parts of the Sketch



The screenshot shows the Arduino IDE interface with the 'Blink' sketch open. The code is divided into three main sections: comments/explanation, setup, and loop.

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

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

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

```
void loop() {
    digitalWrite(13, HIGH);      // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(13, LOW);       // set the LED off
    delay(1000);                // wait for a second
}
```

Comments /
Explaining
the game

Setup /
Stretching or
tying shoes

Loop /
Playing the
game

COMMENTS

- Comments can be inserted ANYWHERE
- Comments are created with // (single line) or /* and */ (multiple lines)
- Comments do not affect the code
- Help others to understand what a particular section of the code does

OPERATORS

- The equals sign
- `=` is used to assign a value
- `==` is used to compare values

OPERATORS

- And & Or
- && is “and”
- || is “or”

Variables

- Basic variable types:
- Boolean
- Integer
- Character

Declaring Variables

- 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 =***

Variable Scope

Where you declare your variables matters

```
/*  
 *  
 * Blink  
 *  
 * Turns on an LED on for one second, then off for one second, repeatedly.  
 *  
 * This example code is in the public domain.  
 */  
  
const int variable1 = 1;  
  
int variable2 = 2;  
  
void setup() {  
    int variable3 = 3;  
  
    // initialize the digital pin as an output  
    // Pin 13 has an LED connected on most Arduino Boards.  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH); // set the LED on  
    delay(1000); // wait for a second  
    digitalWrite(13, LOW); // set the LED off  
    delay(1000); // wait for a second  
}
```

**Constant / Read only
Variable available**

**anywhere
Variable available only
in this function,
between curly brackets**

Setup

void 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

Setup

void 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 header will never change, everything else that occurs in setup

Setup

```
void setup () {  
pinMode (13, OUTPUT); }
```

```
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, now we have to use **CARD**

Setup

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

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

- Serial communication also begins in setup
- This particular example declares Serial communication at the standard baud rate of 9600.

If Statement

if (this is true) { do this; }

```
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);
    }
}
```

If Statement

If

if (this is true) { do this; }

```
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);
  }
}
```

Conditional

if (this is true) { do this; }

```
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);
  }
}
```

Conditional inside
parenthesis,
uses ==, <=, >= or !
you can also nest
using && or ||

Statement(s)

if (this is true) { do this; }

```
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);
  }
}
```

Action that occurs if
conditional is true,
inside of curly brackets,
can be anything,
even more if statements

Else

else { do this; }

```
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

LOOP

void loop () {}

Blink



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

This example code is in the public domain.
*/



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

void loop() {
    digitalWrite(13, HIGH);                                // set the LED on
    delay(1000);                                         // wait for a second
    digitalWrite(13, LOW);                                 // set the LED off
    delay(1000);                                         // wait for a second
}
```

Loop

LOOP

void loop () {}

```
Blink
```

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

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

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

void loop() {
    digitalWrite(13, HIGH);      // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(13, LOW);       // set the LED off
    delay(1000);                // wait for a second
}
```

Loop header

LOOP

void loop () {}

- The “void” in the header is what the function will return (or spit out) when it happens, in this case it returns nothing so it is void

LOOP

*void **loop** () {}*

- The “loop” in the header is what the function is called, sometimes you make the name up, sometimes (like loop) the function already has a name

LOOP

void loop () {}

- The “()” in the header is where you declare any variables that you are “passing” (or sending) the function, the loop function is never “passed” any variables

LOOP

void loop () {}

Blink



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

This example code is in the public domain.
*/



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

void loop() {
    digitalWrite(13, HIGH);      // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(13, LOW);       // set the LED off
    delay(1000);                // wait for a second
}
```

**Loop body
between curly
brackets**

Basic Repetition

- For
- While

Basic Repetition

```
for (int count = 0; count<10; count++)
{
    //for action code goes here
    //this could be anything
}
```

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for(int i = 0; i < 8; i++){                                //This is a loop and will run
        pinMode(ledPins[i],OUTPUT); //we use this to set each LED pin
    }                                                       //the code this replaces is
    /* (commented code will not run)
     * these are the lines replaced by the for loop above they do exactly the same thing the one above just uses less typing
     */
    pinMode(ledPins[0],OUTPUT);
    pinMode(ledPins[1],OUTPUT);
    pinMode(ledPins[2],OUTPUT);
}
```

For loop

Basic Repetition

for (int count = 0; count<10; count++)

{

//for action code goes here

}

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for(int i = 0; i < 8; i++){
        pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
    }                                //the code this replaces is

    /* (commented code will not run)
     * these are the lines replaced by the for loop above they do e:
     * same thing the one above just uses less typing
    pinMode(ledPins[0],OUTPUT);
    pinMode(ledPins[1],OUTPUT);
    pinMode(ledPins[2],OUTPUT);
    pinMode(ledPins[3],OUTPUT);
}
```

For header

Basic Repetition

```
for (int count = 0; count<10; count++)
{
    //for action code goes here
}
```

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for int i = 0; i < 0; i++){ //This is a loop and will run
        pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
    } //the code this replaces is

    /* (commented code will not run)
     * these are the lines replaced by the for loop above they do e:
     * same thing the one above just uses less typing
     pinMode(ledPins[0],OUTPUT);
     pinMode(ledPins[1],OUTPUT);
     pinMode(ledPins[2],OUTPUT);
     pinMode(ledPins[3],OUTPUT);
    
```

Basic Repetition

for (*int count = 0; count<10; count++*)

{

//for action code goes here

}

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for(int i = 0; i < 0; i++) {
        pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
    }                                //the code that replaces is
                                         Declare a variable
                                         and assign it a
                                         value
/* (commented code will not run)
 * these are the lines replaced by the for loop above they do e:
 * same thing the one above just uses less typing
pinMode(ledPins[0],OUTPUT);
pinMode(ledPins[1],OUTPUT);
pinMode(ledPins[2],OUTPUT);
pinMode(ledPins[3],OUTPUT);
```

Basic Repetition

```
for (int count = 0; count<10; count++)
{
    //for action code goes here
}
```

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for(int i = 0; i < 8; i++){
        pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
    }

    /* (commented code will not run)
     * these are the lines replaced by the for loop above. they do e:
     * same thing the one above just uses less typing
    pinMode(ledPins[0],OUTPUT);
    pinMode(ledPins[1],OUTPUT);
    pinMode(ledPins[2],OUTPUT);
    pinMode(ledPins[3],OUTPUT);
}
```

If this conditional
is true do the code
inside the curly
brackets, if it's
false the computer
exits the for loop

Basic Repetition

for (int count = 0; count<10; count++)

{

//for action code goes here

}

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for(int i = 0; i < 8; i++) {i++ Change variable  
so the computer  
isn't stuck inside  
for loop forever
        pinMode(ledPins[i], OUTPUT); //we use this to set each LED p
    }

    /* (commented code will not run)
     * these are the lines replaced by the for loop above them to e:
     * same thing the one above just uses less typing
    pinMode(ledPins[0], OUTPUT);
    pinMode(ledPins[1], OUTPUT);
    pinMode(ledPins[2], OUTPUT);
    pinMode(ledPins[3], OUTPUT);
```

Basic Repetition

for (int count = 0; count<10; count++)

{

//for action code goes here

}

```
void setup()
{
    //Set each pin connected to an LED to output mode (pulling high
    for(int i = 0; i < 8; i++){
        pinMode(ledPins[i], OUTPUT) //this is a loop and will re
    }                                //we use the standard for loop
    //the code this replaces is
    /* (commented code will not run)
     * these are the lines replaced by the for loop above they do e:
     * same thing the one above just uses less typing
    pinMode(ledPins[0], OUTPUT);
    pinMode(ledPins[1], OUTPUT);
    pinMode(ledPins[2], OUTPUT);
    pinMode(ledPins[3], OUTPUT);
```

**Curly brackets
contain the for
loop body code**

**Code that occurs
each time the for
loop repeats**

Basic Repetition

```
while ( count<10 )
```

```
{
```

```
//while action code goes here
```

```
//should include a way to change count
```

```
//variable so the computer is not stuck
```

```
//inside the while loop forever
```

```
}
```

Basic Repetition

```
int count=0;  
while ( count<10 )  
{  
//looks basically like a “for” loop  
//except the variable is declared before  
//and incremented inside the while  
//loop  
}
```

Digital Read

```
while ( digitalRead(buttonPin)==1 )
{
  //instead of changing a variable
  //you just read a pin so the computer
  //exits when you press a button
  //or a sensor is tripped
}
```

Digital Write

***void setup () {
digitalWrite (12, HIGH); }***

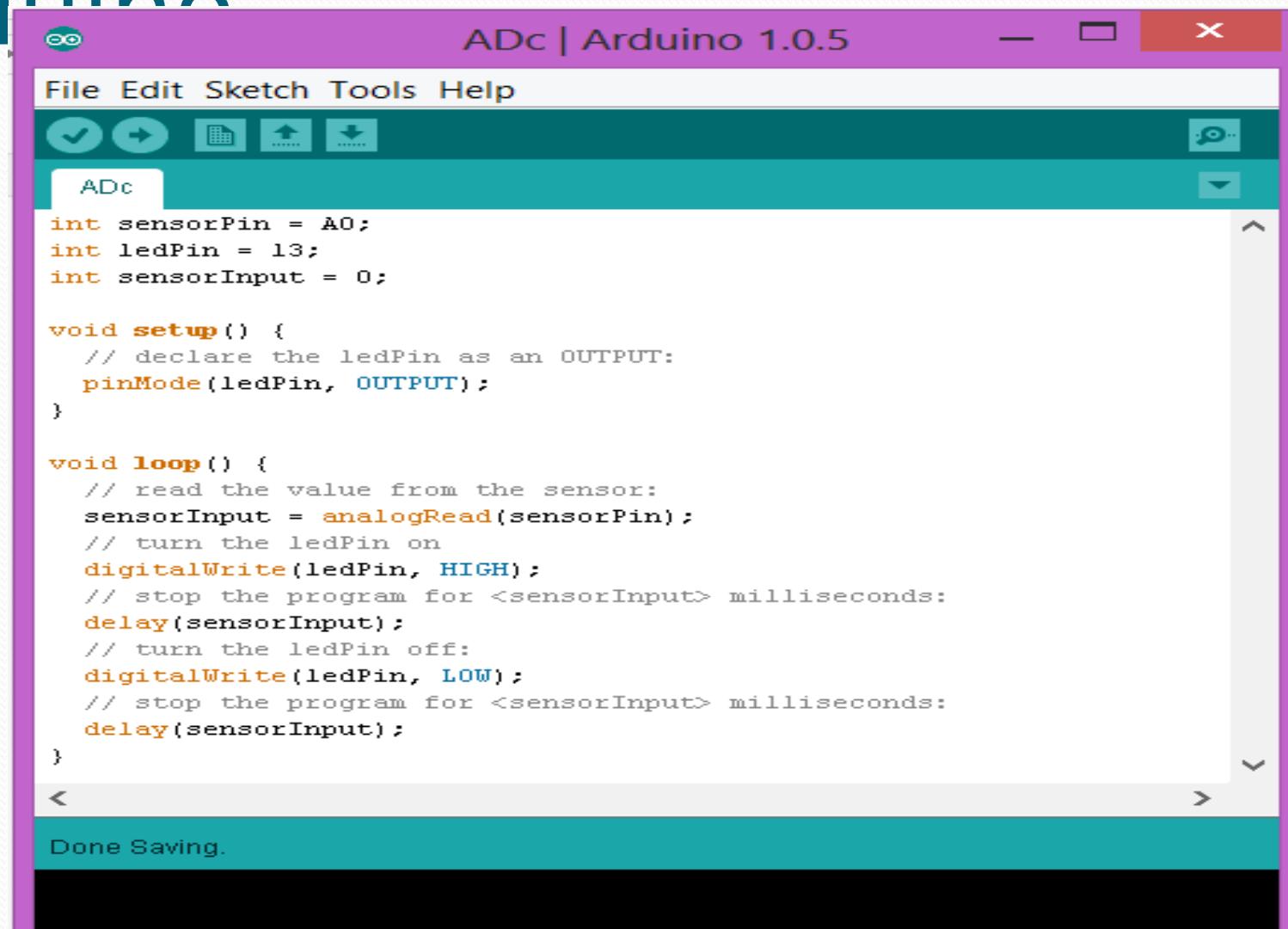
```
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
  Serial.begin(9600);  
  digitalWrite(12, HIGH);  
}
```

- `digitalWrite` is used to give output values to DIGITAL pins, i.e., HIGH voltage or LOW voltage.

Analog to Digital Conversion in Arduino

- Analog to Digital Conversion simply means you get an analog input and give a digital output.
- During the conversion, some error is introduced due to approximation of analog value to closest digital value.

Analog to Digital Conversion in Arduino



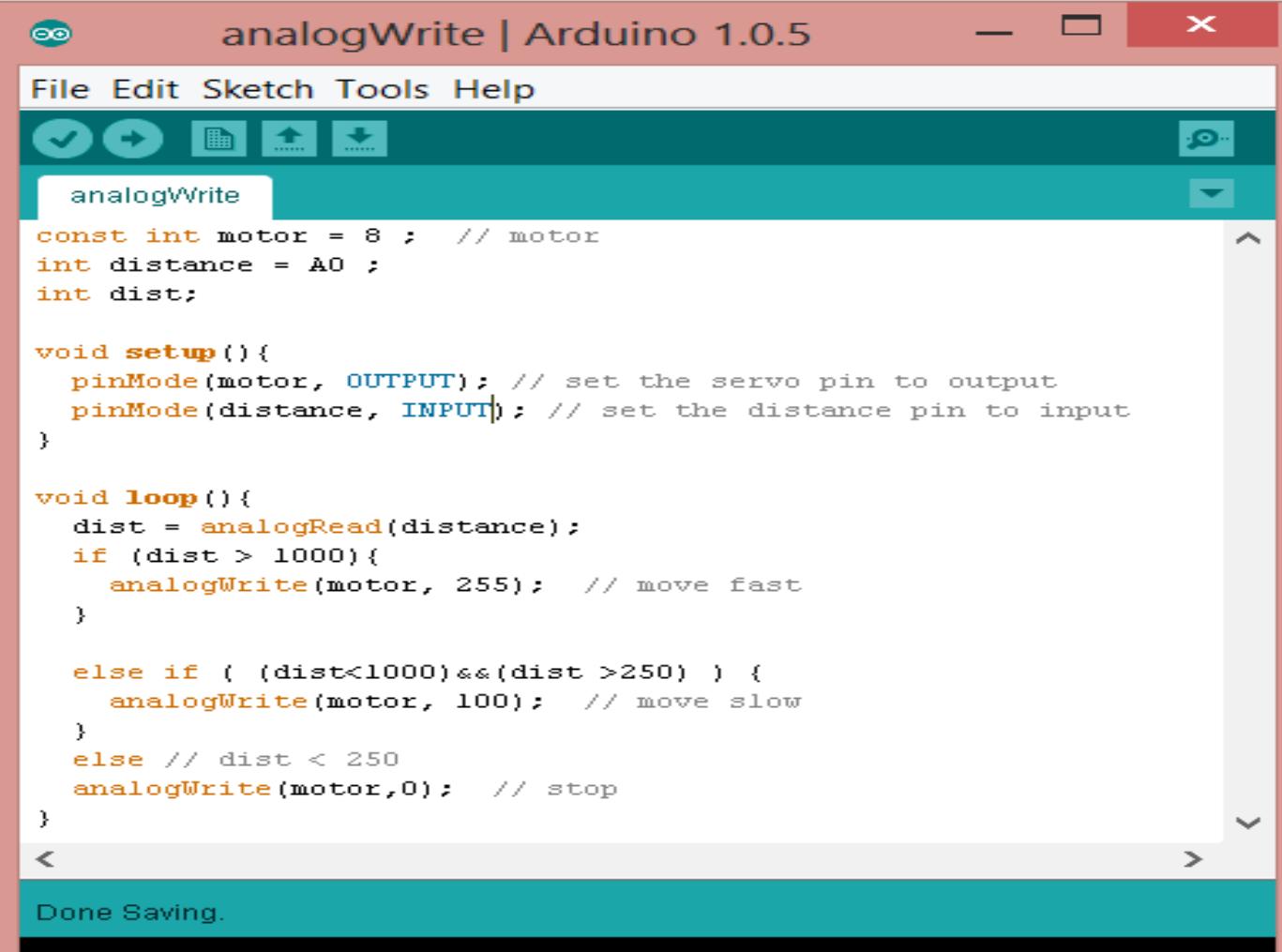
The screenshot shows the Arduino IDE interface with a sketch named "ADC". The code implements analog-to-digital conversion by reading the value from an analog pin (A0) and then turning on an LED connected to digital pin 13 for a duration proportional to the sensor input value.

```
ADc | Arduino 1.0.5
File Edit Sketch Tools Help
ADC
int sensorPin = A0;
int ledPin = 13;
int sensorInput = 0;

void setup() {
    // declare the ledPin as an OUTPUT:
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // read the value from the sensor:
    sensorInput = analogRead(sensorPin);
    // turn the ledPin on
    digitalWrite(ledPin, HIGH);
    // stop the program for <sensorInput> milliseconds:
    delay(sensorInput);
    // turn the ledPin off:
    digitalWrite(ledPin, LOW);
    // stop the program for <sensorInput> milliseconds:
    delay(sensorInput);
}
< Done Saving. >
```

Analog Write



The screenshot shows the Arduino IDE interface with a sketch named "analogWrite". The code uses analogRead() to read distance from pin A0 and analogWrite() to move a motor based on the distance value.

```
analogWrite | Arduino 1.0.5
File Edit Sketch Tools Help
analogWrite
const int motor = 8 ; // motor
int distance = A0 ;
int dist;

void setup(){
  pinMode(motor, OUTPUT); // set the servo pin to output
  pinMode(distance, INPUT); // set the distance pin to input
}

void loop(){
  dist = analogRead(distance);
  if (dist > 1000){
    analogWrite(motor, 255); // move fast
  }

  else if ( (dist<1000)&&(dist >250) ) {
    analogWrite(motor, 100); // move slow
  }
  else // dist < 250
    analogWrite(motor,0); // stop
}
```

Done Saving.

Showing readings on Serial Monitor

The image shows the Arduino IDE interface. On the left is the Serial Monitor window titled '/dev/ttyACM0' with a 'Send' button. It displays the value '1023' repeated 20 times. At the bottom are settings for 'Autoscroll', 'No line ending', and '9600 baud'. On the right is the Sketch Editor window titled 'sketch_feb20a | Arduino 1.0'. The code reads:

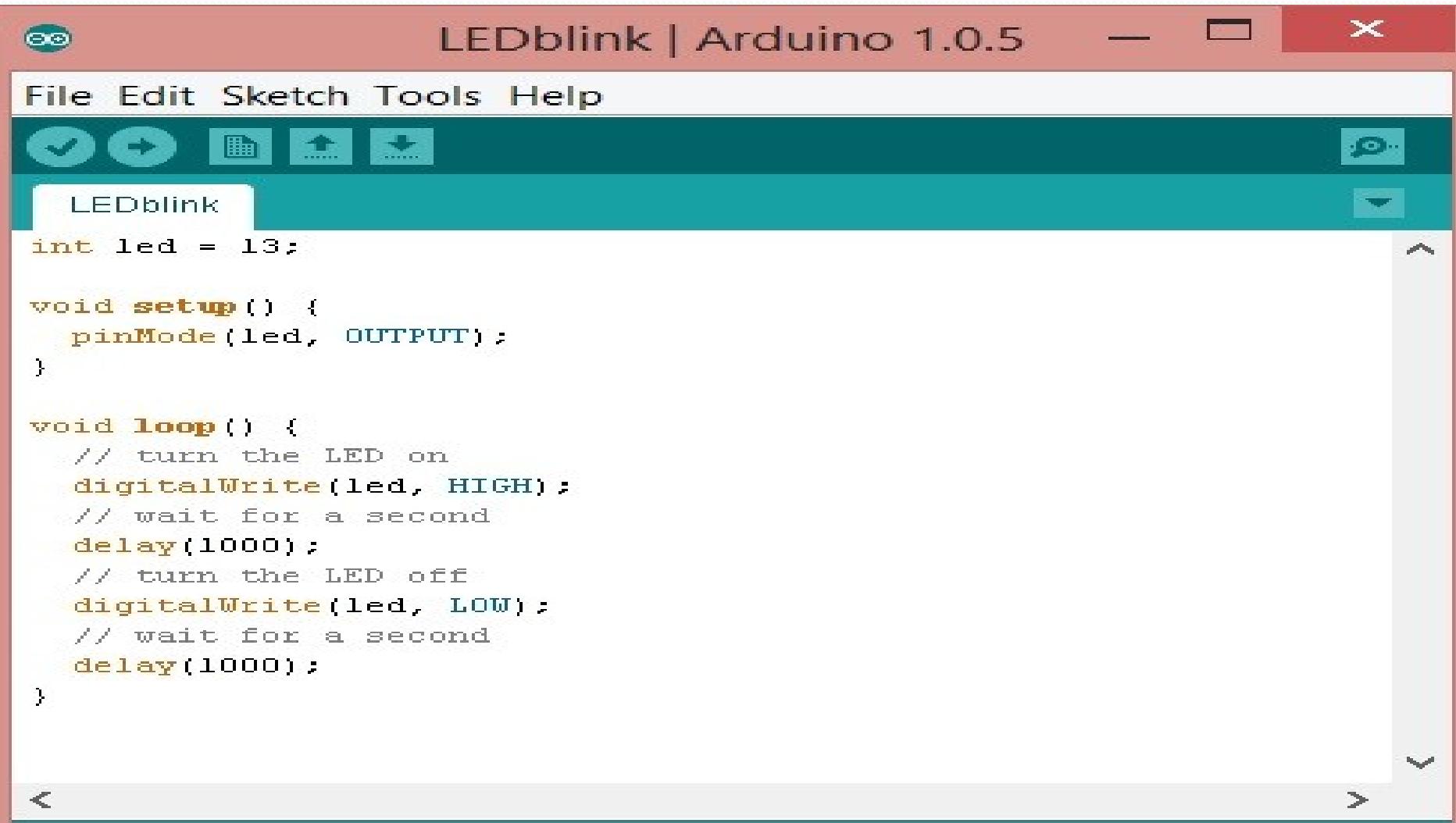
```
sketch_feb20a
File Edit Sketch Tools Help
sketch_feb20a

//One time setup
void setup() {
    //Serial communication to be establish at 9600 baud rate
    Serial.begin(9600);
}

//Loop runs again over and over
void loop() {
    // create an integer variable name valueDigital and name it as
    int value = analogRead(A3);
    // print out the value you read:
    Serial.println(value);
    // delay to let arduino read and send data to serial terminal.
    delay(1000);
}
```

The status bar at the bottom indicates 'Arduino Mega 2560 or Mega ADK on /dev/ttyACM0'.

Program for blinking an LED



The screenshot shows the Arduino IDE interface with the title bar "LEDblink | Arduino 1.0.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for file operations. The main window displays the "LEDblink" sketch code:

```
int led = 13;

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

void loop() {
  // turn the LED on
  digitalWrite(led, HIGH);
  // wait for a second
  delay(1000);
  // turn the LED off
  digitalWrite(led, LOW);
  // wait for a second
  delay(1000);
}
```



Questions ?

Thanks!

