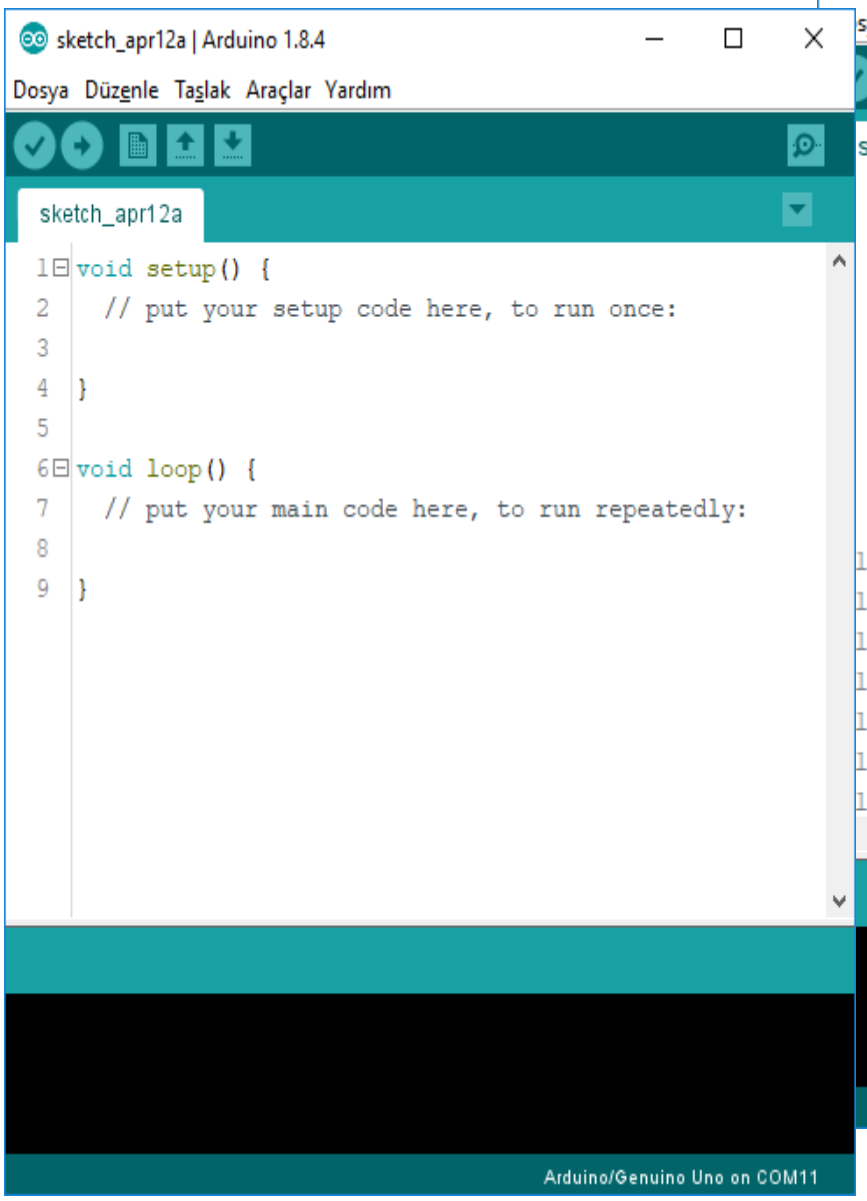
# Arduino: programming

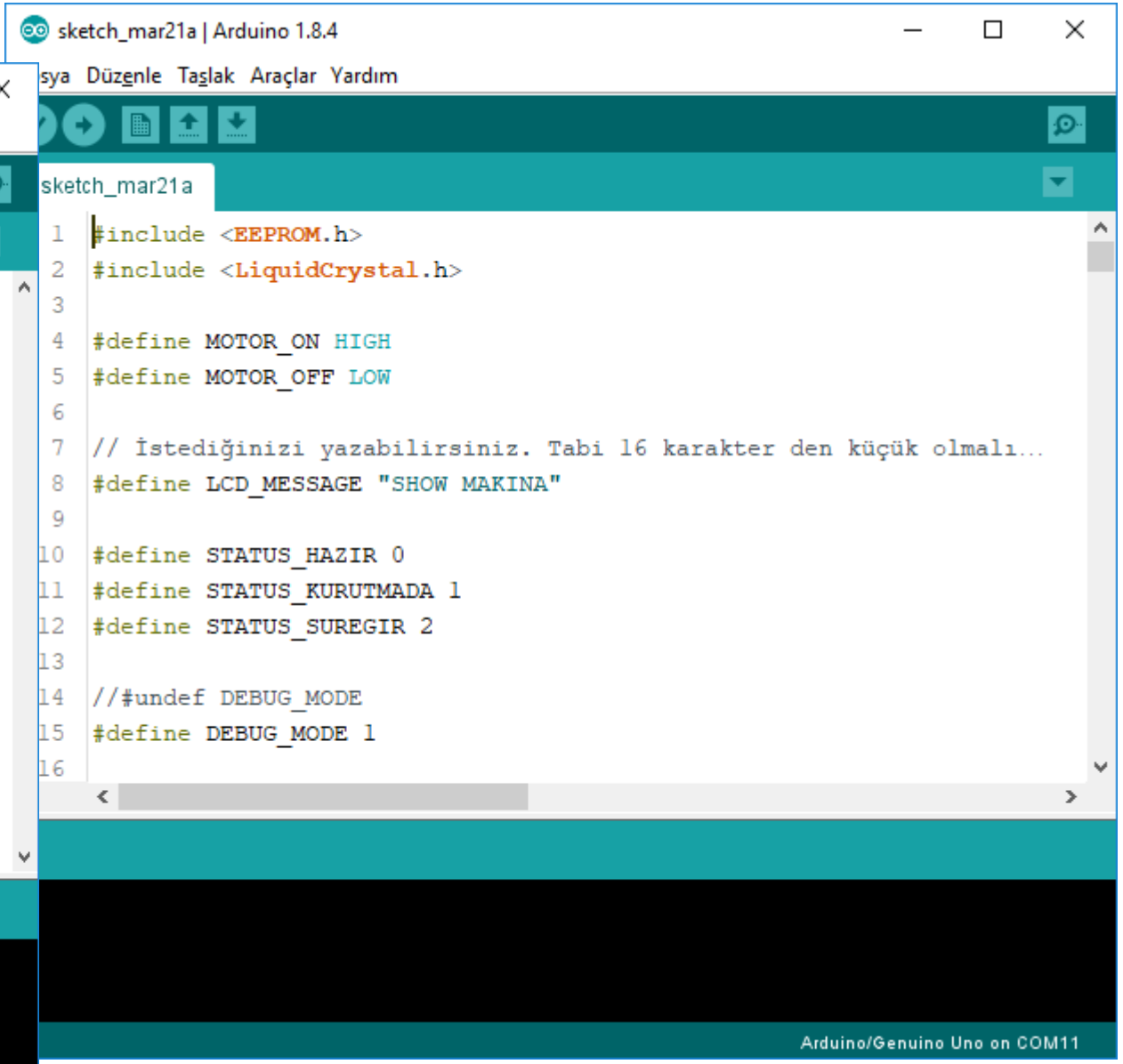
Asst. Prof. Dr. Yalçın İŞLER

# Textbooks

- https://www.tutorialspoint.com/arduino/index.htm
- https://maker.robotistan.com/etiket/arduino-dersleri/

# Arduino IDE


sketch_apr12a | Arduino 1.8.4
Dosya Düzenle Taslak Araçlar Yardım
sketch_apr12a
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8
9 }
Arduino/Genuino Uno on COM11


sketch_mar21a | Arduino 1.8.4
sya Düzenle Taslak Araçlar Yardım
sketch_mar21a
1 #include <EEPROM.h>
2 #include <LiquidCrystal.h>
3
4 #define MOTOR_ON HIGH
5 #define MOTOR_OFF LOW
6
7 // İstediğinizi yazabilirsiniz. Tabi 16 karakter den küçük olmalı...
8 #define LCD_MESSAGE "SHOW MAKINA"
9
10 #define STATUS_HAZIR 0
11 #define STATUS_KURUTMADA 1
12 #define STATUS_SUREGIR 2
13
14 //#undef DEBUG_MODE
15 #define DEBUG_MODE 1
16
Arduino/Genuino Uno on COM11

# Data types

- void boolean char unsigned char byte int
- unsigned int word long unsigned long short float
- double array String (char array and/or object)

```
#define STATUS_HAZIR 0

boolean val = false ;

int iStatus = STATUS_HAZIR;

int const EEPROM_MOTOR_ILERI = 0;

unsigned char a = 'a';

unsigned char a = 65;
```

# Variable scope

- **GLOBAL VARIABLE:** Outside of all functions, which is called global variables.
- **LOCAL VARIABLE:** Inside a function or a block, which is called local variables.
- **FORMAL PARAMETER:** In the definition of function parameters, which is called formal parameters.

- Braces “ { and } “ are used to indicate blocks of code or flow control.

# Identifiers

- An identifier is a name used to identify a variable, constant, function, or other objects.
- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Arduino does not allow punctuation characters such as @, $, +, -, ?, *, and % within identifiers.
- Arduino is a case sensitive programming language. Thus, x and X are different. Similarly, loop() and Loop() are different.

# Naming convention

- Constants and definitions are all uppercase  letter.
- Starting an identifier with a single or two leading underscore indicates that the identifier is compiler directive.
- If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.
- All other identifiers start with a lowercase letter.
- If the identifier consists of more words, the following words will start with a capital letter (pinMode).
- I prefer adding the variable type as a first letter of the variable (int iTemp; boolean bGirisVarMi; long lUzunluk).

# Commenting

- A double division sign (//) that is not inside a string literal is the beginning of a comment. All characters after the //, up to the end of the physical line, are part of the comment and the Arduino ignores them:

```
// First comment
PinMode(13, OUTPUT); // 13 nolu pin cikis olsun
```

- Arduino supports multiple-line commenting feature between /* and */ .

```
/*
  First demo program.
  Yalcin Isler @2019.
*/
```

# Blank lines

- A line containing only whitespace, possibly with a comment, is known as a blank line and Arduino totally ignores it.
- Every command should be ended with ;
- Definitions and some special commands do not require ending with ;

# Arithmetic & Comparison operators

- addition + subtraction - multiplication * division / modulus %

- equal == not equal != greater > less < greater-equal >= less-equal <=

# Assignment & Bitwise & Logical operators

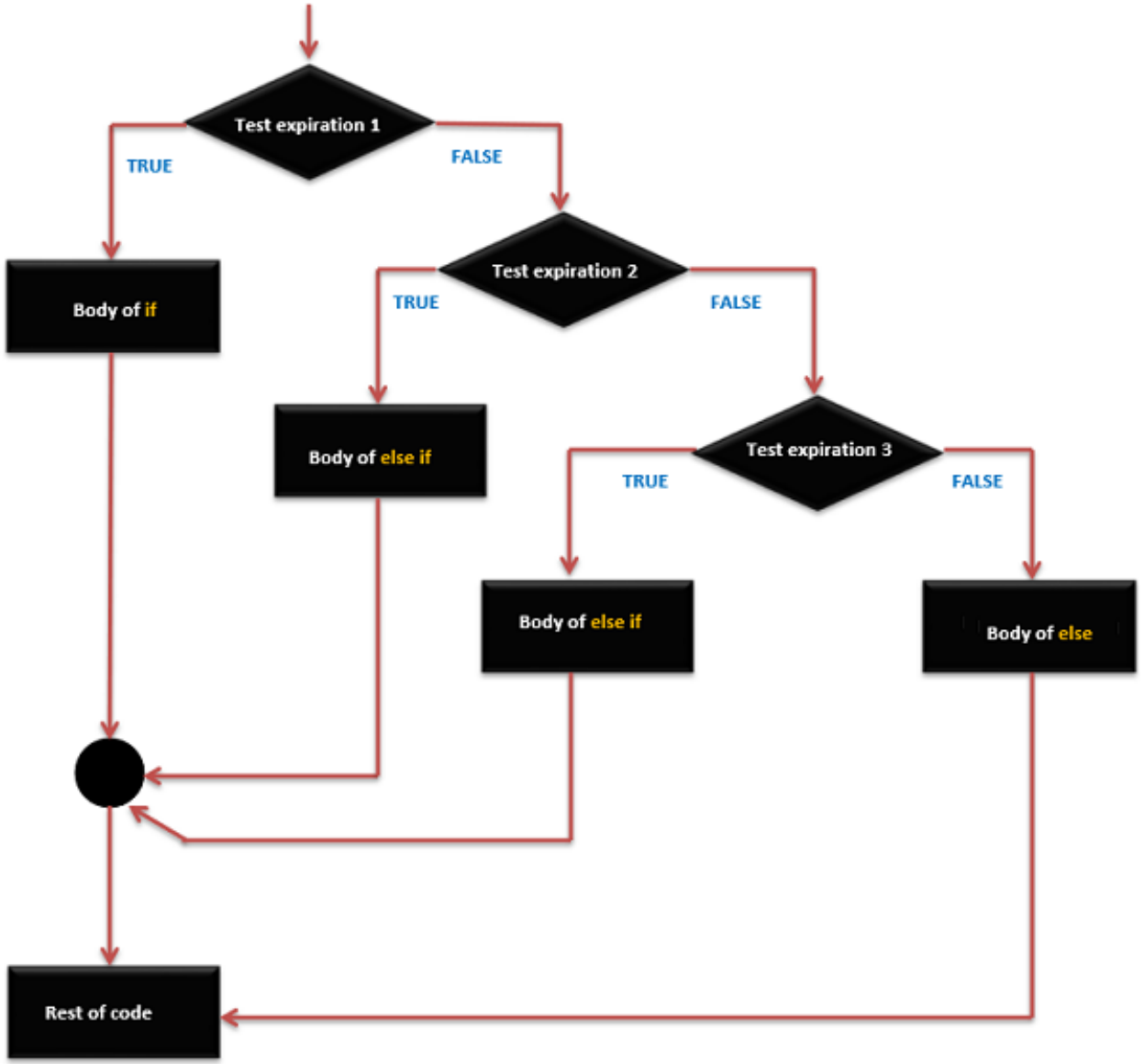
- = += -= *= /= %= &= |=

- And & Or | Ex-or ^
- Not (1’s complement) ~
- Shift-Left << Shift-Right >>

- X && Y X || Y ! X

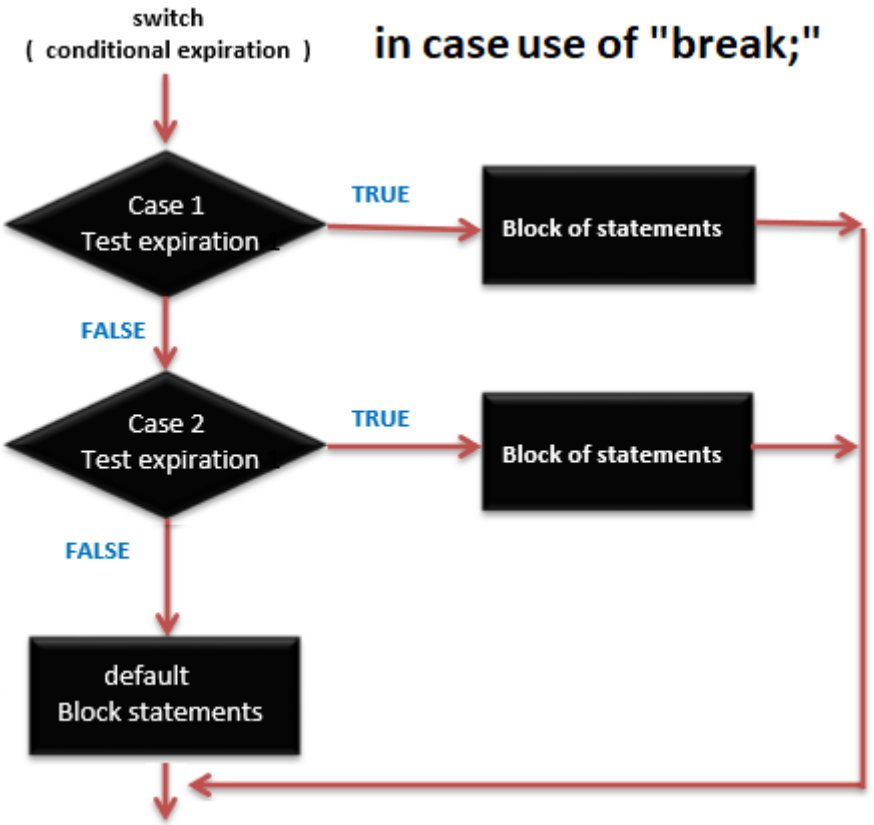
# Conditional: if - else if - else

```
if (expression_1) {
   Block of statements;
}
else if (expression_2) {
   Block of statements;
}
...
else {
   Block of statements;
}
```



# Conditional: switch - case

```
switch (variable) {
   case label1:
   // statements
   break;
case label2:
   // statements
   break;
default:
   // statements
   break;
}
```



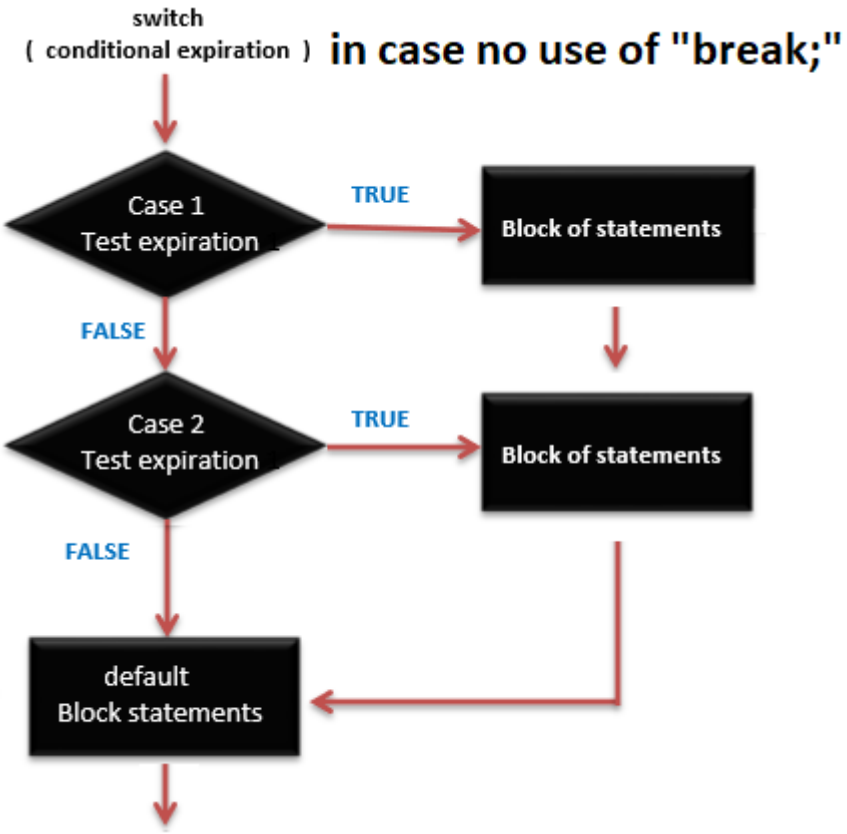
# Conditional: switch - case

```
switch (variable) {
   case label1:
   // statements

case label2:
   // statements

default:
   // statements

}
```



# Conditional: ?

expression1 ? expression2 : expression3

- expression1 must be a scalar expression.
- Both expression2 and expression3 have to be of arithmetic type.
- Both expression2 and expression3 are subjected to usual arithmetic conversions, which determines the resulting type.

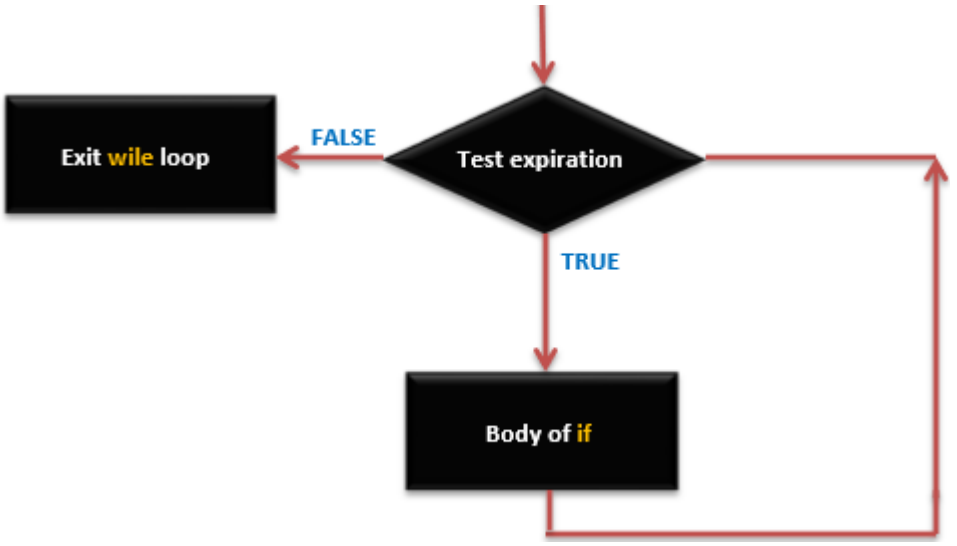
```
/* Find max(a, b): */

max = ( a > b ) ? a : b;
```

# Loop: while

```
while(expression) {
   Block of statements;
}
```

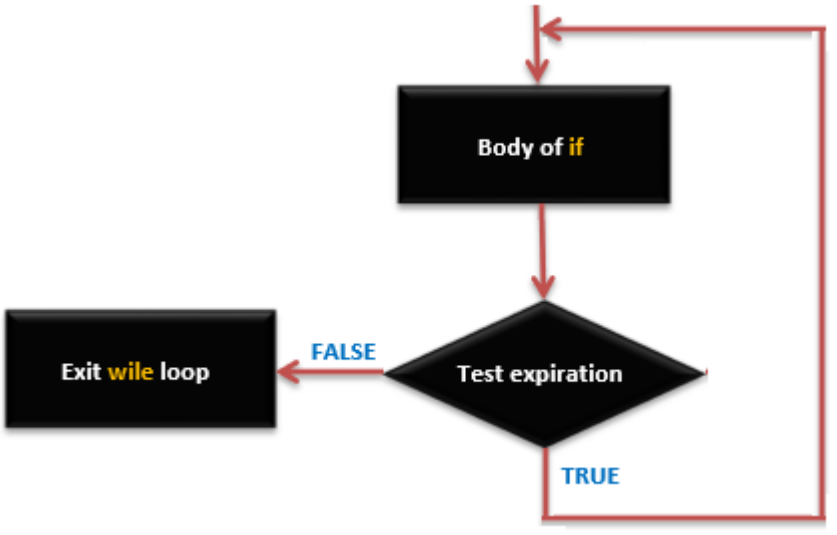
- while loops will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. Something must change the tested variable, or the while loop will never exit.



# Loop: do-while

```
do {
   Block of statements;
} while(expression);
```

- Similar to the while loop, but the block of statements will run before testing the expression.

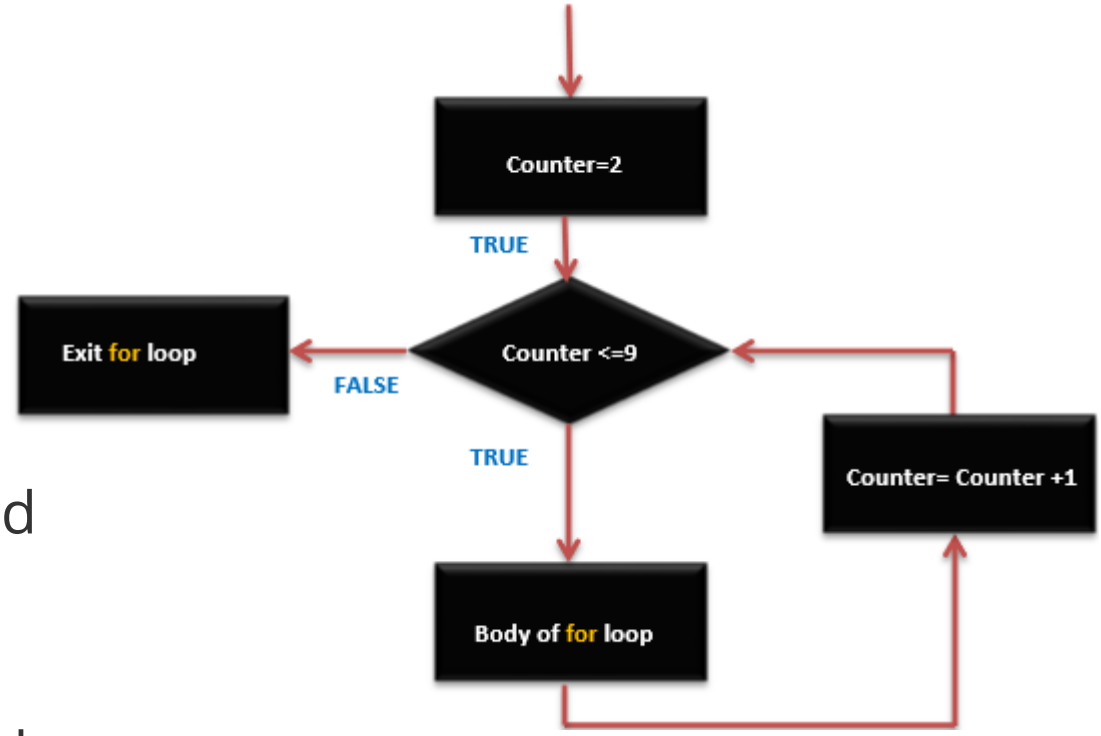


# Loop: for

```
for ( initialize; control;
increment or decrement)
{
   // statement block
}
```

- A for loop executes statements a predetermined number of times. The control expression for the loop is initialized, tested and manipulated entirely within the for loop parentheses.

```
for(counter = 2;counter <= 9;counter++) {
   //statements block will executed 10 times
}
```



# Infinite (end-less) loops

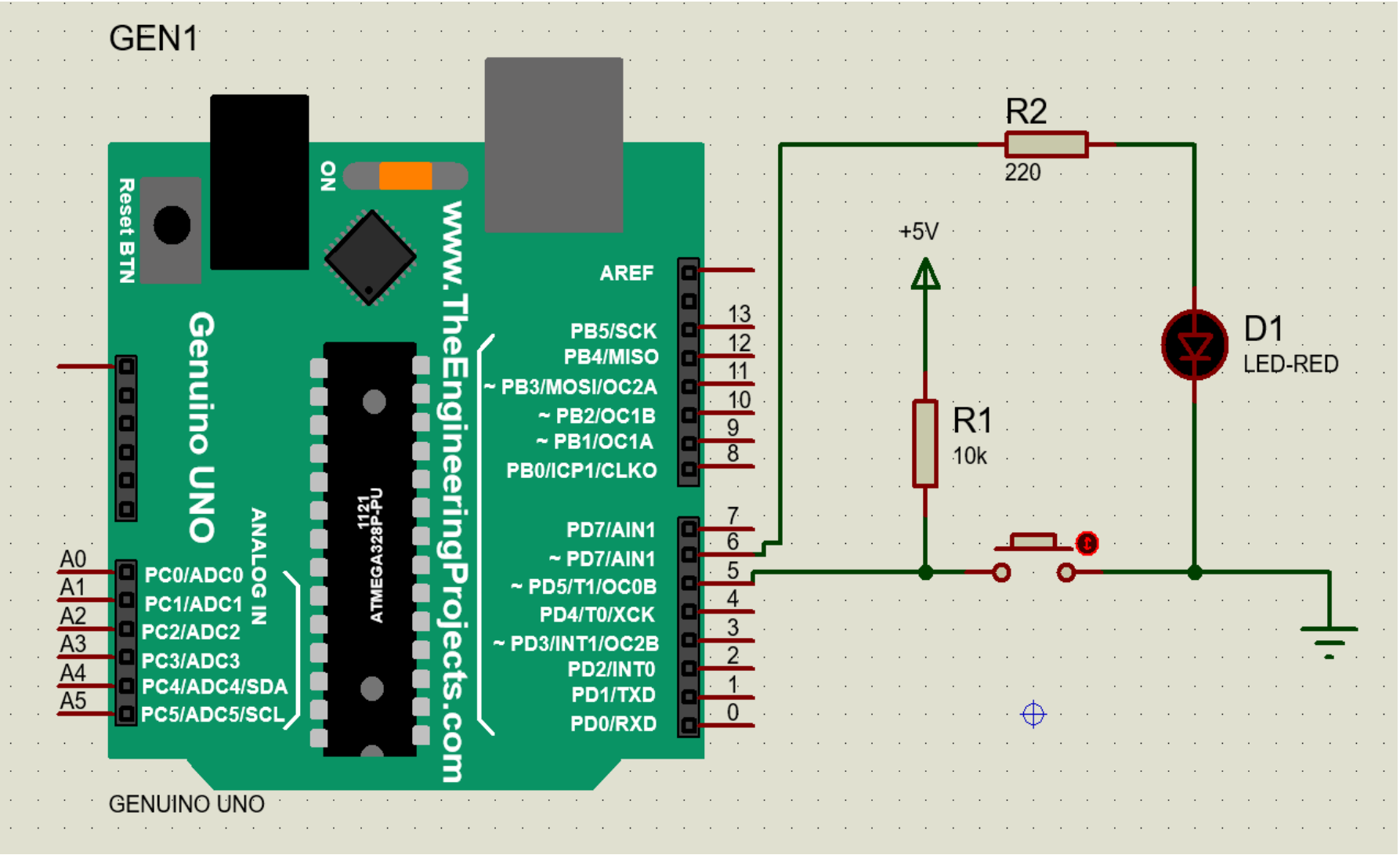
- It is the loop having no terminating condition, so the loop becomes infinite.

```
for (;;) {
   // statement block
}
```

```
do {
   Block of statements;
} while(1);
```

```
while(1) {
   // statement block
}
```

# Simple digital input and output application



# pinMode, digitalRead, digitalWrite, delay

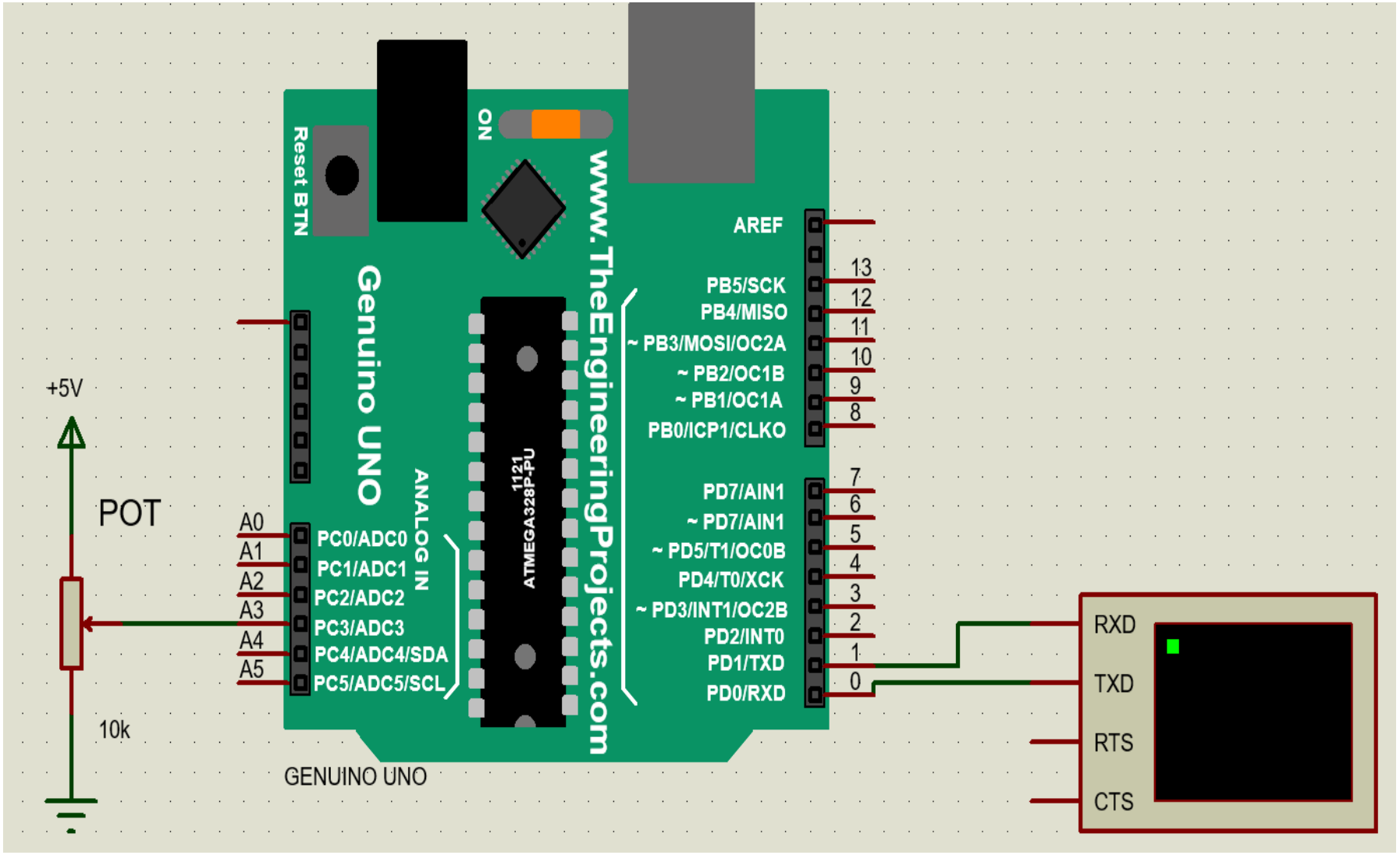
```
int button = 5 ; // button connected to pin 5
int LED = 6; // LED connected to pin 6

void setup () {
   pinMode(button , INPUT_PULLUP);
   // set the digital pin as input with pull-up resistor
   pinMode(button , OUTPUT); // set the digital pin as output
}

void setup () {
   If (digitalRead(button ) == LOW) // if button pressed {
      digitalWrite(LED,HIGH); // turn on led
      delay(500); // delay for 500 ms
      digitalWrite(LED,LOW); // turn off led
      delay(500); // delay for 500 ms
   }
}
```

INPUT, OUTPUT, INPUT_PULLUP, INPUT_PULLDOWN

# Simple analog input and serial port application



# analogRead, Serial.*

```
int analogPin = 3;//potentiometer wiper (middle terminal)
   // connected to analog pin 3
int val = 0; // variable to store the value read

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

void loop() {
   val = analogRead(analogPin); // read the input pin
   Serial.println(val); // debug value
}
```

Digital value of an analog pin may be 0 to 1023.