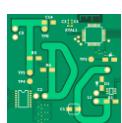


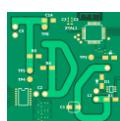
```
void setup() {  
    //runs once on startup  
}  
  
void loop() {  
    //runs again and again  
}
```

	compile program, check for errors
	compile and if successful, upload
	new project
	open
	save
	handling project files (new, rename)
	open serial terminal



Variables

bool, boolean	logical	true, false
byte, unsigned char	8 bit, non negative integer	0-255
char	8 bit, character	'a'
unsigned short	16 bit, non negative integer	0 – 65,535
short	16 bit, integer	-32,768 – 32,767
uint	non negative integer	0 - ?
int	integer	?
unsigned long	at least 32 bit, non negative integer	0 - 4,294,967,295
long	at least 32 bit, integer	-2,147,483,648 - 2,147,483,647
unsigned long long	at least 64 bit, non negative integer	0-18,446,744,073,709,551,615
long long	at least 64 bit, integer	-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807
float	32 bit floating-point fraction	-3.4×10^{38} - 3.4×10^{38}
double	64 bit floating-point fraction	$\pm 5.0 \times 10^{-324}$ - $\pm 1.7 \times 10^{308}$
string, String	text	"hello" "text"



Grammar

making our own function

```
void f(){}
```

void	f	()	{}
return value, what „gives back” (nothing, apple, 42)	name	parameters	body, commands go here

using a function without parameters

```
interrupts();
```

interrupts	()	;
name	parentheses	end of command

using a function with one parameter

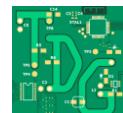
```
delay(42);
```

delay	(42)	;
name	parameter starts here	parameter	parameter end	end of command

using a function with multiple parameters

```
digitalWrite(13, HIGH);
```

digitalWrite	(13	,	HIGH)	;
name	parameters start here	first parameter	separating parameters	second parameter	parameters end	end of command



facebook.com/thedeveloperGuy
youtube.com/thedeveloperGuy

Digital IO

<code>pinMode(pin, mode);</code>	setting the operating mode
<code>INPUT</code>	reading signals
<code>INPUT_PULLUP</code>	reading signals, pullup resistor enables
<code>OUTPUT</code>	writing signals

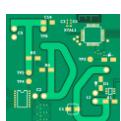
<code>bool digitalRead(pin);</code>	reading the actual value of a pin
<code>digitalWrite(pin, state);</code>	driving IO pin to a voltage
<code>HIGH</code>	system power (3.3V, 5V)
<code>LOW</code>	ground (0V)

Analog IO

<code>int analogRead(pin);</code>	do ADC on selected pin
<code>analogReference(reference);</code>	set ADC reference
<code>analogReadResolution(bit);</code>	ADC resolution
<code>analogWrite(pin, value);</code>	PWM signal, 0-255

Time

<code>unsigned long millis();</code>	millisecs elapsed since start
<code>unsigned long micros();</code>	microsecs elapsed since start
<code>delay(time);</code>	wait (milliseconds)
<code>delayMicroseconds(time);</code>	wait (microseconds)



Structure, class, object

```
class Sensor{
    int pin;
    static int updateCount;
public:
    Sensor();
    Sensor(int pin);
    bool getPin();
    static void update();
};
```

Interrupts

```
void isr(void){
    interrupt handler code
}
attachInterrupt(digitalPinToInterrupt(pin),
    isr, event);
```

<code>digitalPinToInterrupt(pin)</code>	interrupt number from pin
<code>attachInterrupt (interruptNumber, isr, mode);</code>	assign function call to interrupt event
<code>LOW</code>	low signal level
<code>RISING</code>	rising edge (<code>LOW → HIGH</code>)
<code>FALLING</code>	falling edge (<code>HIGH → LOW</code>)
<code>CHANGE</code>	both rising and falling edge
<code>HIGH (Due, Zero, MKR1000)</code>	high signal level

