

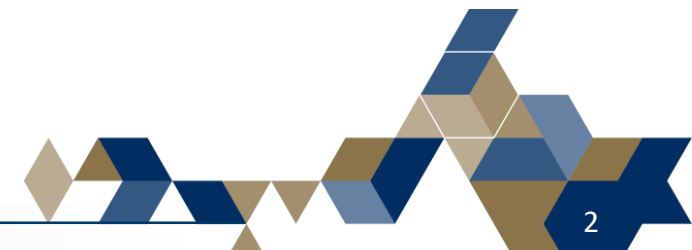
BLG351E-Microcomputer Lab.

Fall 2020-2021

Res. Assist. Kadir OZLEM

General Information

- Presentation days and hours:
 - Monday: 13:30-15:30 , CRN11641,
 - Thursday: 13:30-15:30 , CRN11639,
 - Friday: 15:30-17:30 , CRN11637.
- Course Web Page: Ninova path: BLG351/CRN: <<Your CRN Number>>
- Instructor: İlkey Öksüz
- Teaching Assistants:
 - Abdullah Ekrem Okur (okurabd@itu.edu.tr)
 - Kadir Özlem (kadir.ozlem@itu.edu.tr)
 - Talip Tolga Sarı (sarita@itu.edu.tr)
 - Tuğba Pamay Arslan (pamay@itu.edu.tr)



Evaluation Criteria

- Laboratory performance: 50% (8 Experiments x 6.25 points),
 - %25 Code & Design
 - %25 Presentation
- Reports: 50% (8 experiments x 6.25 points)
- Note:
 - You will make experiments as groups that consist of 3 students.
 - If a student didn't join a group and not on any list, they will be inserted into other groups randomly.
 - If your group has less than 3 students, other students will join your group randomly.
 - Only one student in the group should send the list through the Ninova.



Schedule

Week*	Activity	Homework
19.10.2020	No Class	
26.10.2020	No Class	
02.11.2020	Introduction to the Laboratory	
09.11.2020		Experiment 1
16.11.2020	Experiment 1	Experiment 2
23.11.2020	Experiment 2	Experiment 3
30.11.2020	Experiment 3	Experiment 4
07.12.2020	Experiment 4	Experiment 5
14.12.2020	Experiment 5	Experiment 6
21.12.2020	Experiment 6	
28.12.2020	No Class (Happy New Year)	Experiment 7
04.01.2021	Experiment 7	Experiment 8
11.01.2021	Experiment 8	
18.01.2021	No Class	

* Week that starts on the shown date

- Presentation
- Homework

Arduino Board

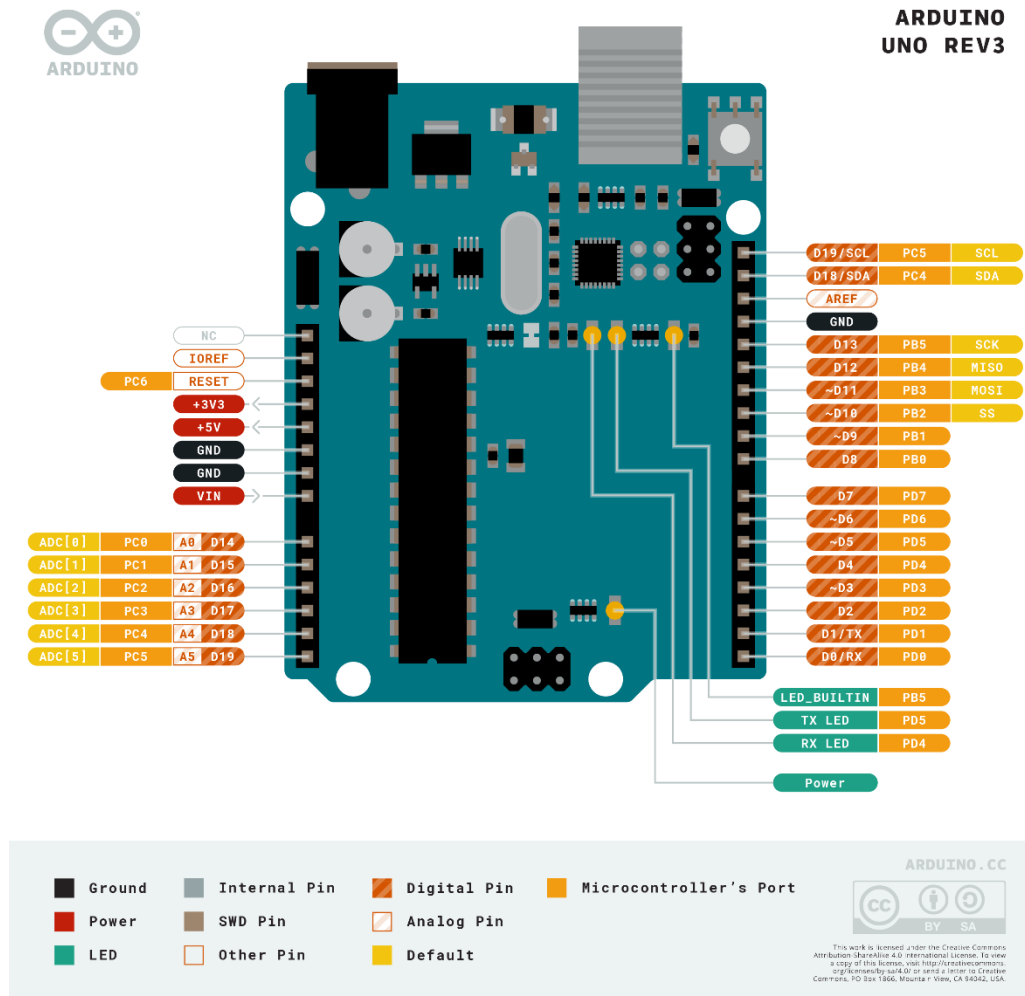


AUTODESK®
TINKERCAD®

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- Arduino boards are able to **read inputs** - light on a sensor, a finger on a button, or a Twitter message - and **turn it into an output** - activating a motor, turning on an LED, publishing something online
- Arduino Uno Board (REV 3) will be used.
- **Tinkercad** is a free, easy-to-use app for 3D design, electronics, and coding.
- Experiments will be in Arduino programming language.
- (Experiments contains assembly code blocks)
- Architecture of the Arduino UNO is important.

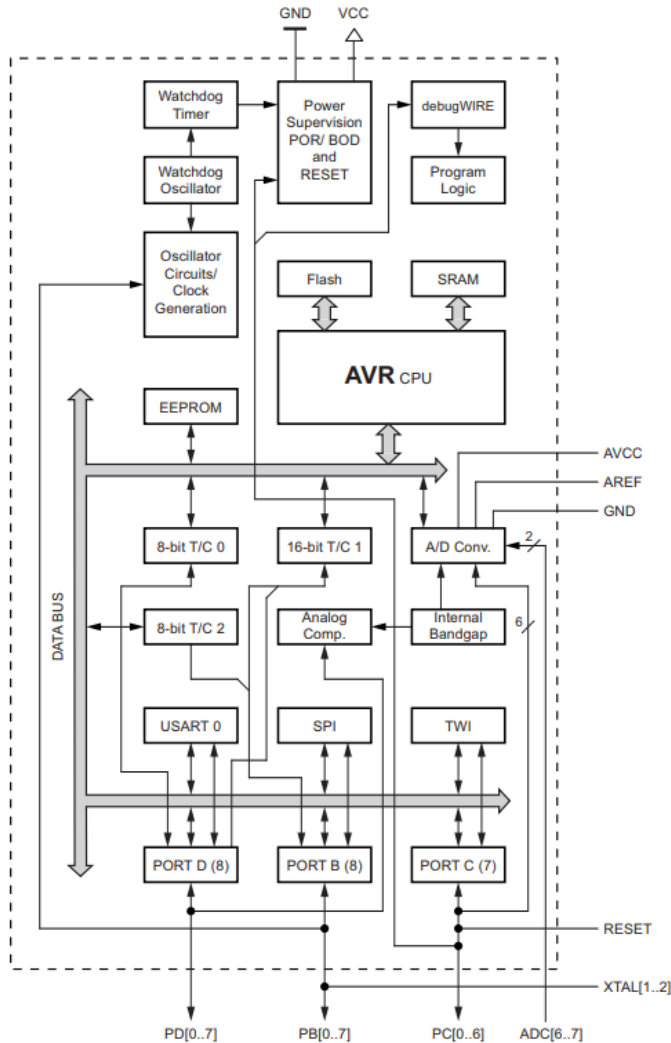


Arduino UNO Specifications

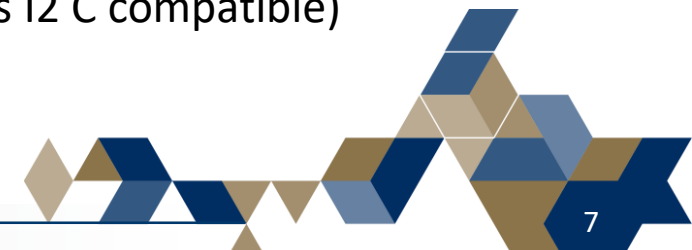


Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

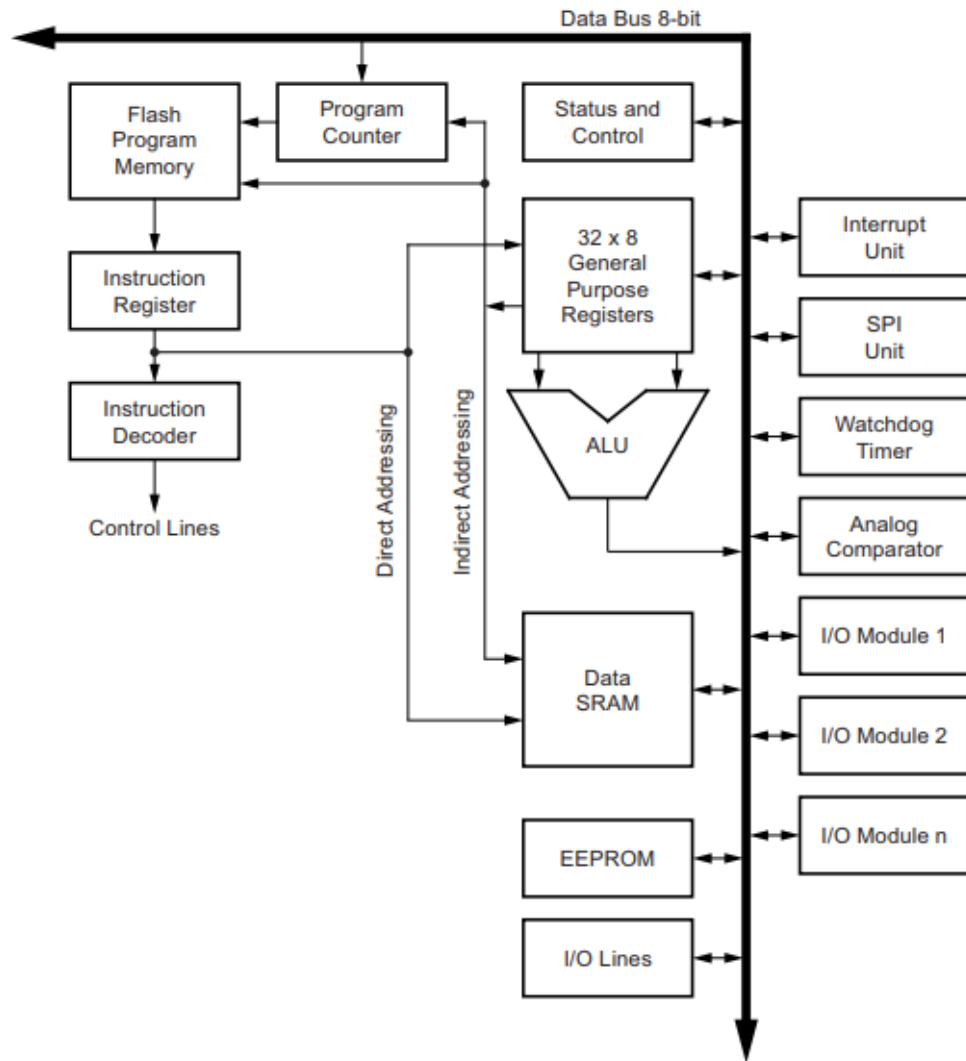
Atmel® ATmega328P



- A low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture
- High endurance non-volatile memory segments
 - 32K bytes of in-system self-programmable flash program memory
 - 1Kbytes EEPROM
 - 2Kbytes internal SRAM
- Peripheral features
 - 23 programmable I/O lines
 - Two 8-bit Timer/Counters and One 16-bit Timer/Counter
 - Real time counter with separate oscillator
 - Six PWM channels
 - 8-channel 10-bit ADC
 - Programmable serial USART
 - Master/slave SPI serial interface
 - Byte-oriented 2-wire serial interface (Phillips I2 C compatible)
 - On-chip analog comparator
 - Interrupt and wake-up on pin change



Atmel® ATmega328P



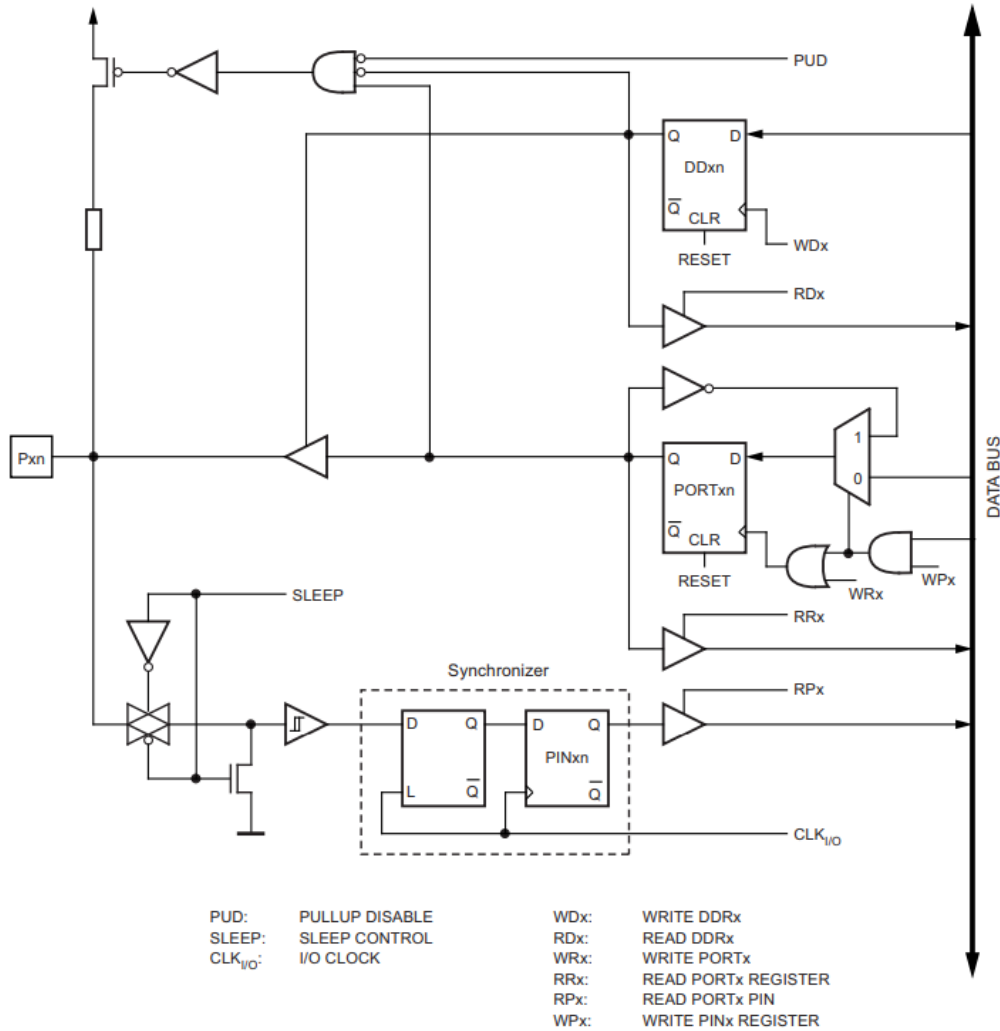
The AVR status register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

General
Purpose
Working
Registers

7	0	Addr.	
R0		0x00	
R1		0x01	
R2		0x02	
...			
R13		0x0D	
R14		0x0E	
R15		0x0F	
R16		0x10	
R17		0x11	
...			
R26		0x1A	X-register Low Byte
R27		0x1B	X-register High Byte
R28		0x1C	Y-register Low Byte
R29		0x1D	Y-register High Byte
R30		0x1E	Z-register Low Byte
R31		0x1F	Z-register High Byte

Atmel® ATmega328P – General Digital I/O



MCUCR – MCU Control Register

[illegible]

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

PINB – The Port B Input Pins Address

[illegible]

Arduino Programming Language

- Includes.
- Global variable declaration.
- Initialize variables, pin modes, start using libraries.
- Only run once, after each powerup or reset.
- Loops consecutively
- Allow your program to change and respond
- Use it to actively control the Arduino board.

```

1  volatile uint8_t* port;
2
3  void setup()
4  {
5      DDRB |= B00000001;
6      port = &PORTB;
7      Serial.begin(115200);
8  }
9
10 void loop()
11 {
12     PORTB |= B00000001;
13     Serial.println(*port);
14     delay(1000); // Wait for 1000 millisecond(s)
15     PORTB &= B11111110;
16     Serial.println(*port);
17     delay(1000); // Wait for 1000 millisecond(s)
18 }
  
```



Arduino Programming Language

FUNCTIONS

For controlling the Arduino board and performing computations.

Digital I/O

digitalRead()
digitalWrite()
pinMode()

Analog I/O

analogRead()
analogReference()
analogWrite()

Zero, Due & MKR Family

analogReadResolution()
analogWriteResolution()

Advanced I/O

noTone()
pulseIn()
pulseInLong()
shiftIn()
shiftOut()
tone()

Time

delay()
delayMicroseconds()
micros()
millis()

Math

abs()
constrain()
map()
max()
min()
pow()
sq()
sqrt()

Trigonometry

cos()
sin()
tan()

Characters

isAlpha()
isAlphaNumeric()
isAscii()
isControl()
isDigit()
isGraph()
isHexadecimalDigit()
isLowerCase()
isPrintable()
isPunct()
isSpace()
isUpperCase()
isWhitespace()

Random Numbers

random()
randomSeed()

Bits and Bytes

bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

External Interrupts

attachInterrupt()
detachInterrupt()

Interrupts

interrupts()
noInterrupts()

Communication

Serial
Stream

USB

Keyboard
Mouse

VARIABLES

Arduino data types and constants.

Constants

HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED_BUILTIN
true | false
Floating Point Constants
Integer Constants

Conversion

(unsigned int)
(unsigned long)
byte()
char()
float()
int()
long()
word()

Data Types

array
bool
boolean
byte
char
double
float
int
long
short
size_t
string
String()
unsigned char
unsigned int
unsigned long
void
word

Variable Scope & Qualifiers

const
scope
static
volatile

Utilities

PROGMEM
sizeof()

Arduino Programming Language

STRUCTURE

The elements of Arduino (C++) code.

Sketch

`loop()`
`setup()`

Control Structure

`break`
`continue`
`do...while`
`else`
`for`
`goto`
`if`
`return`
`switch...case`
`while`

Further Syntax

`#define` (define)
`#include` (include)
`/* */` (block comment)
`//` (single line comment)
`;` (semicolon)
`{}` (curly braces)

Arithmetic Operators

`%` (remainder)
`*` (multiplication)
`+` (addition)
`-` (subtraction)
`/` (division)
`=` (assignment operator)

Comparison Operators
`!=` (not equal to)
`<` (less than)
`<=` (less than or equal to)
`==` (equal to)
`>` (greater than)
`>=` (greater than or equal to)

Boolean Operators

`!` (logical not)
`&&` (logical and)
`||` (logical or)

Pointer Access Operators

`&` (reference operator)
`*` (dereference operator)

Bitwise Operators

`&` (bitwise and)
`<<` (bitshift left)
`>>` (bitshift right)
`^` (bitwise xor)
`|` (bitwise or)
`~` (bitwise not)

Compound Operators

`%=` (compound remainder)
`&=` (compound bitwise and)
`*=` (compound multiplication)
`++` (increment)
`+=` (compound addition)
`--` (decrement)
`-=` (compound subtraction)
`/=` (compound division)
`^=` (compound bitwise xor)
`|=` (compound bitwise or)

Libraries

The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from **Sketch > Import Library**.

A number of libraries come installed with the IDE, but you can also download or create your own. See [these instructions](#) for details on installing libraries. There's also a [tutorial on writing your own libraries](#). See the API Style Guide for information on making a good Arduino-style API for your library.

- [Communication](#) (754)
- [Data Processing](#) (164)
- [Data Storage](#) (94)
- [Device Control](#) (551)
- [Display](#) (319)
- [Other](#) (287)
- [Sensors](#) (664)
- [Signal Input/Output](#) (262)
- [Timing](#) (146)
- [Uncategorized](#) (128)

Standard Libraries

- [EEPROM](#) - reading and writing to "permanent" storage
- [Ethernet](#) - for connecting to the internet using the Arduino Ethernet Shield, Arduino Ethernet Shield 2 and Arduino Leonardo ETH
- [Firmata](#) - for communicating with applications on the computer using a standard serial protocol.
- [GSM](#) - for connecting to a GSM/GPRS network with the GSM shield.
- [LiquidCrystal](#) - for controlling liquid crystal displays (LCDs)
- [SD](#) - for reading and writing SD cards
- [Servo](#) - for controlling servo motors
- [SPI](#) - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- [SoftwareSerial](#) - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- [Stepper](#) - for controlling stepper motors
- [TFT](#) - for drawing text, images, and shapes on the Arduino TFT screen
- [WiFi](#) - for connecting to the internet using the Arduino WiFi shield
- [Wire](#) - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

Arduino Programming Language

Port Registers

Port registers allow for lower-level and faster manipulation of the i/o pins of the microcontroller on an Arduino board. The chips used on the Arduino board (the ATmega8 and ATmega168) have three ports:

- B (digital pin 8 to 13)
- C (analog input pins)
- D (digital pins 0 to 7)

Each port is controlled by three registers, which are also defined variables in the arduino language. The DDR register, determines whether the pin is an INPUT or OUTPUT. The PORT register controls whether the pin is HIGH or LOW, and the PIN register reads the state of INPUT pins set to input with pinMode(). The maps of the [ATmega8](#) and [ATmega168](#) chips show the ports. The newer Atmega328p chip follows the pinout of the ATmega168 exactly.

DDR and PORT registers may be both written to, and read. PIN registers correspond to the state of inputs and may only be read.

PORTD maps to Arduino digital pins 0 to 7

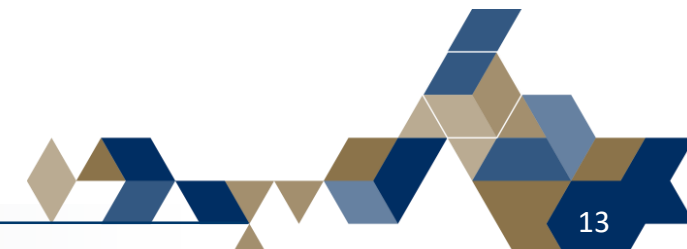
- DDRD - The Port D Data Direction Register - read/write
- PORTD - The Port D Data Register - read/write
- PIND - The Port D Input Pins Register - read only

PORTB maps to Arduino digital pins 8 to 13 The two high bits (6 & 7) map to the crystal pins and are not usable

- DDRB - The Port B Data Direction Register - read/write
- PORTB - The Port B Data Register - read/write
- PINB - The Port B Input Pins Register - read only

PORTC maps to Arduino analog pins 0 to 5. Pins 6 & 7 are only accessible on the Arduino Mini

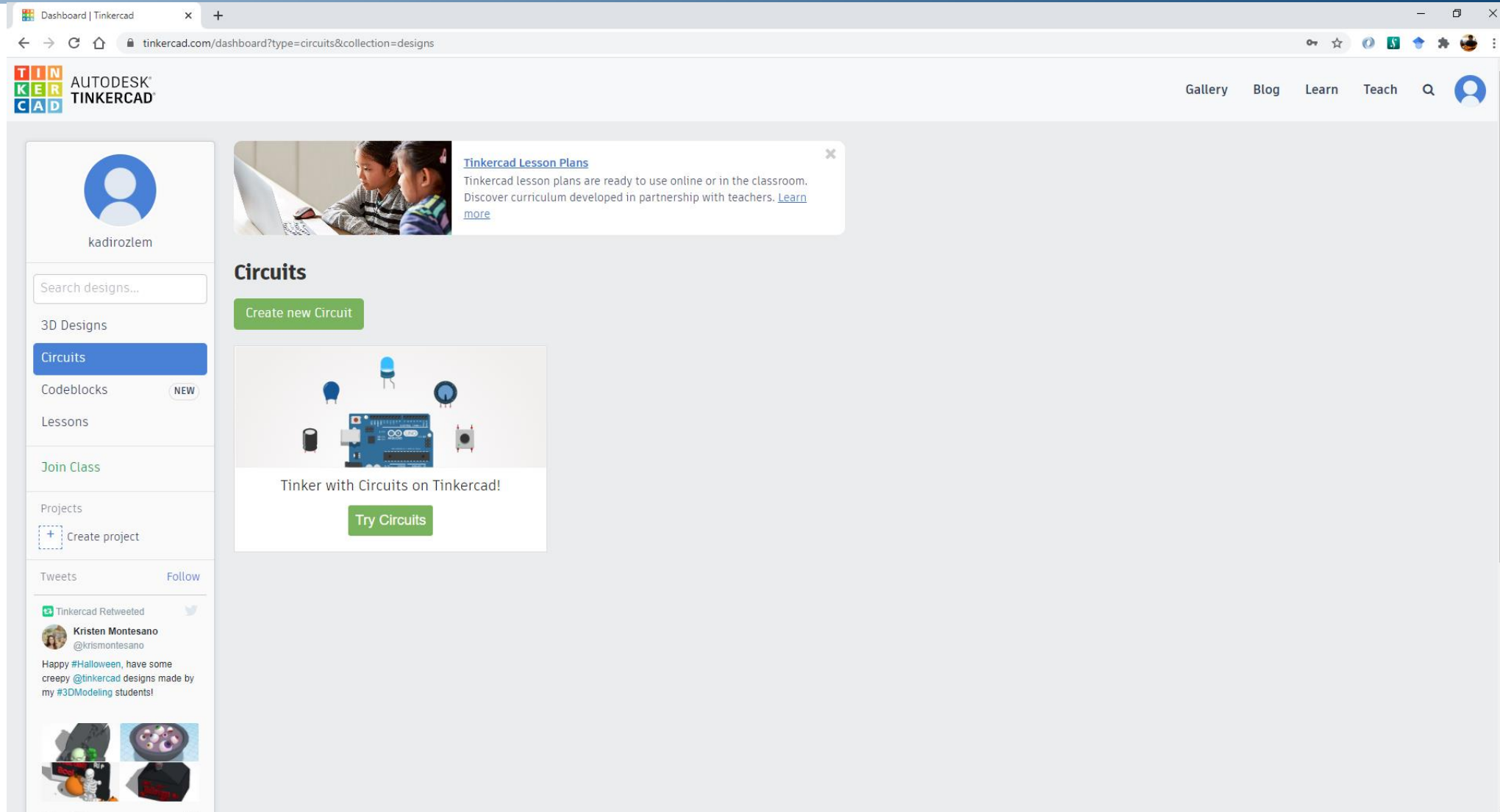
- DDRC - The Port C Data Direction Register - read/write
- PORTC - The Port C Data Register - read/write
- PINC - The Port C Input Pins Register - read only



Demonstration

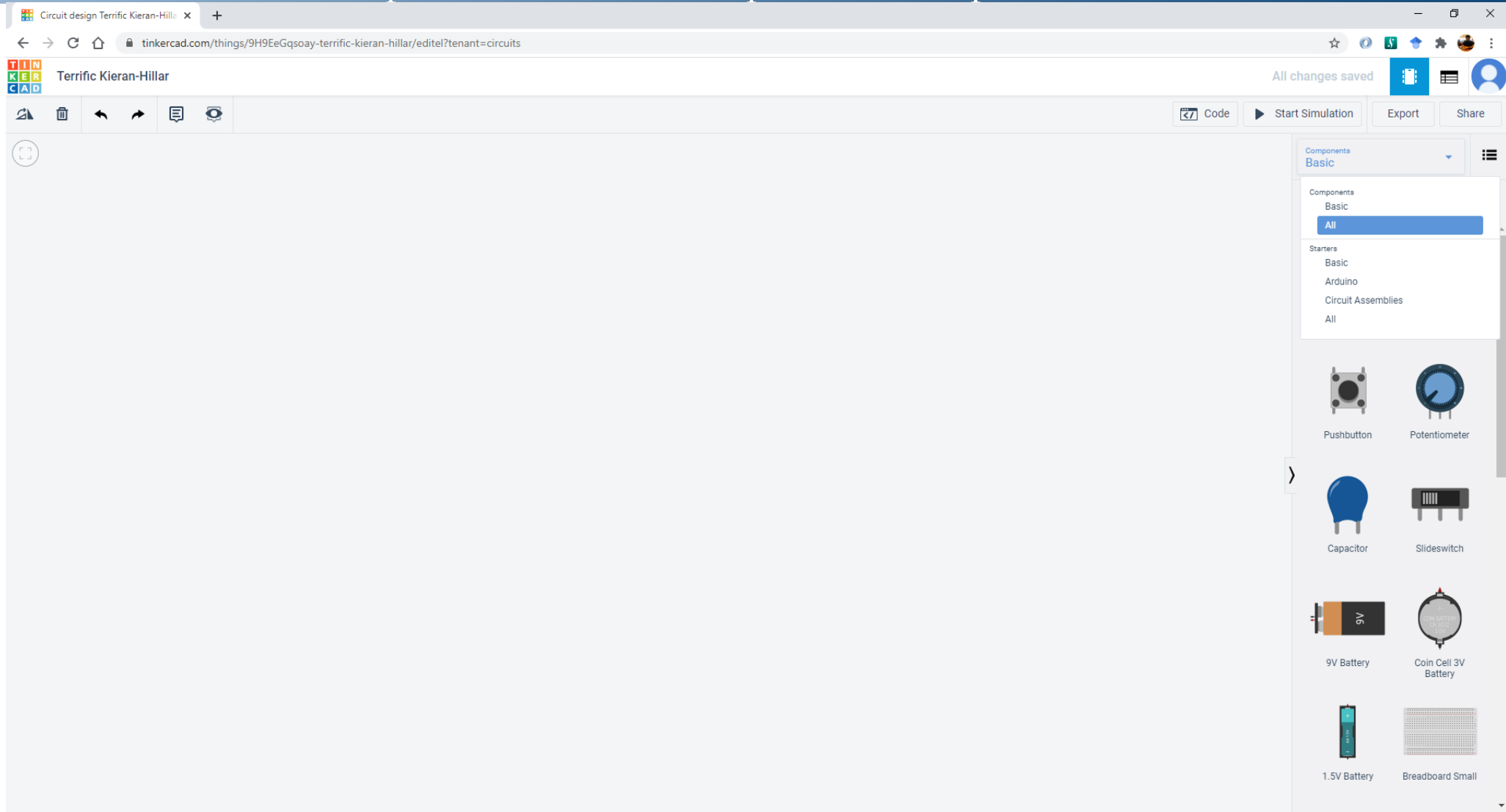


Tinkercad – Click the «Create new Circuits» button

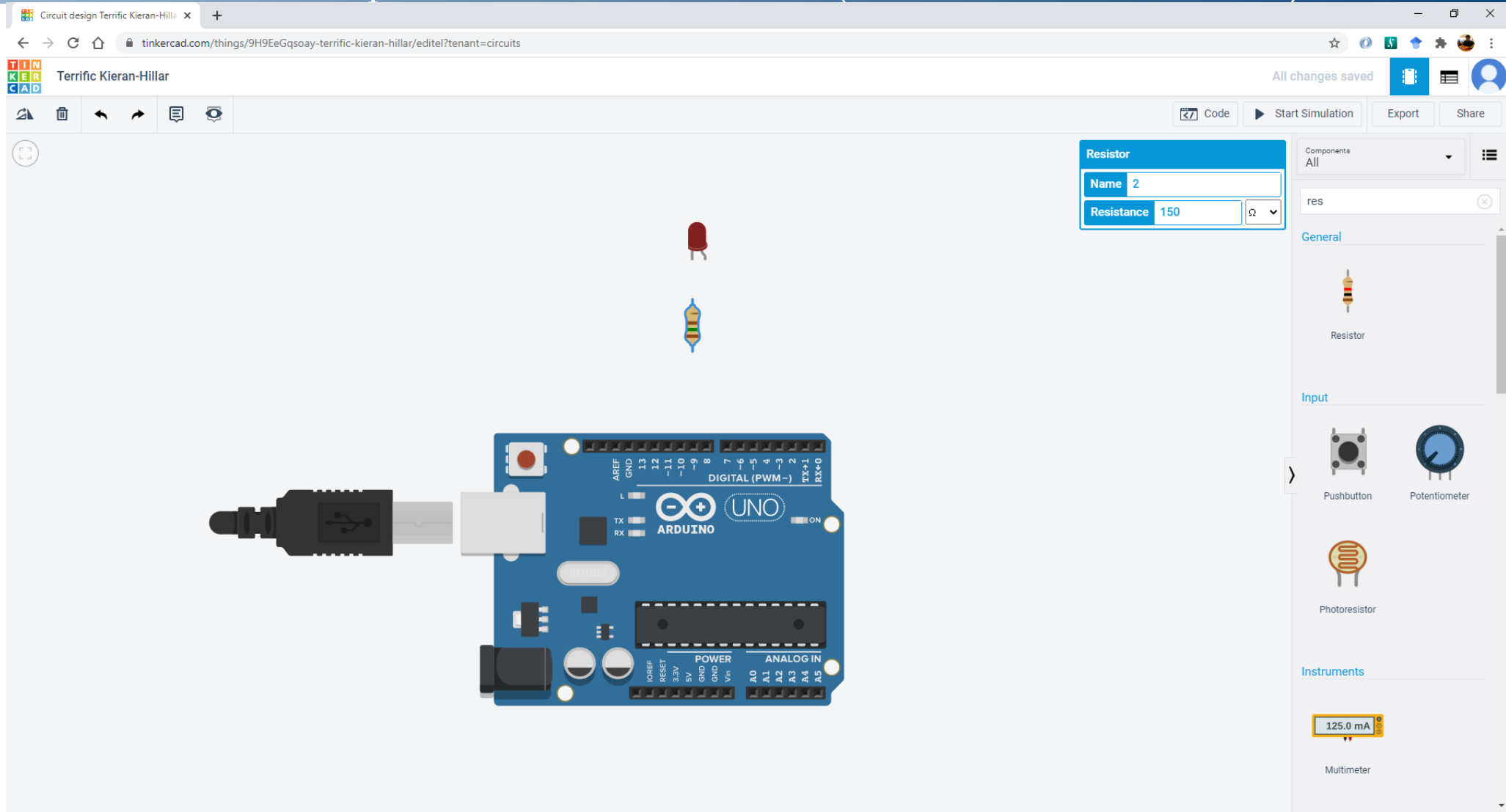


The screenshot shows the Tinkercad dashboard in a web browser. The browser's address bar displays the URL: `tinkercad.com/dashboard?type=circuits&collection=designs`. The dashboard features a sidebar on the left with navigation options: "3D Designs", "Circuits" (highlighted in blue), "Codeblocks" (marked with a "NEW" badge), "Lessons", "Join Class", "Projects" (with a "+ Create project" button), and "Tweets" (with a "Follow" button). The main content area is titled "Circuits" and includes a green "Create new Circuit" button. Below this, there is a promotional graphic for "Tinker with Circuits on Tinkercad!" featuring a breadboard and electronic components, with a green "Try Circuits" button. A "Tinkercad Lesson Plans" banner is visible at the top right of the main content area, stating: "Tinkercad lesson plans are ready to use online or in the classroom. Discover curriculum developed in partnership with teachers. [Learn more](#)". The sidebar also shows a user profile for "kadirozlem" and a tweet from "Kristen Montesano" (@krismontesano) about Halloween and Tinkercad designs.

Tinkercad – Select the All Components under Components Drop-Down Menu



Tinkercad – Add suitable components and set its value (Ex:150 ohm for resistor)



The screenshot shows the Tinkercad web interface with an Arduino Uno circuit. A red LED is connected to a 150 ohm resistor, which is then connected to the Arduino's digital pins. A multimeter is also connected to the circuit, displaying a reading of 125.0 mA. The right-hand panel shows the component selection menu with the Resistor component selected and its value set to 150 ohms.

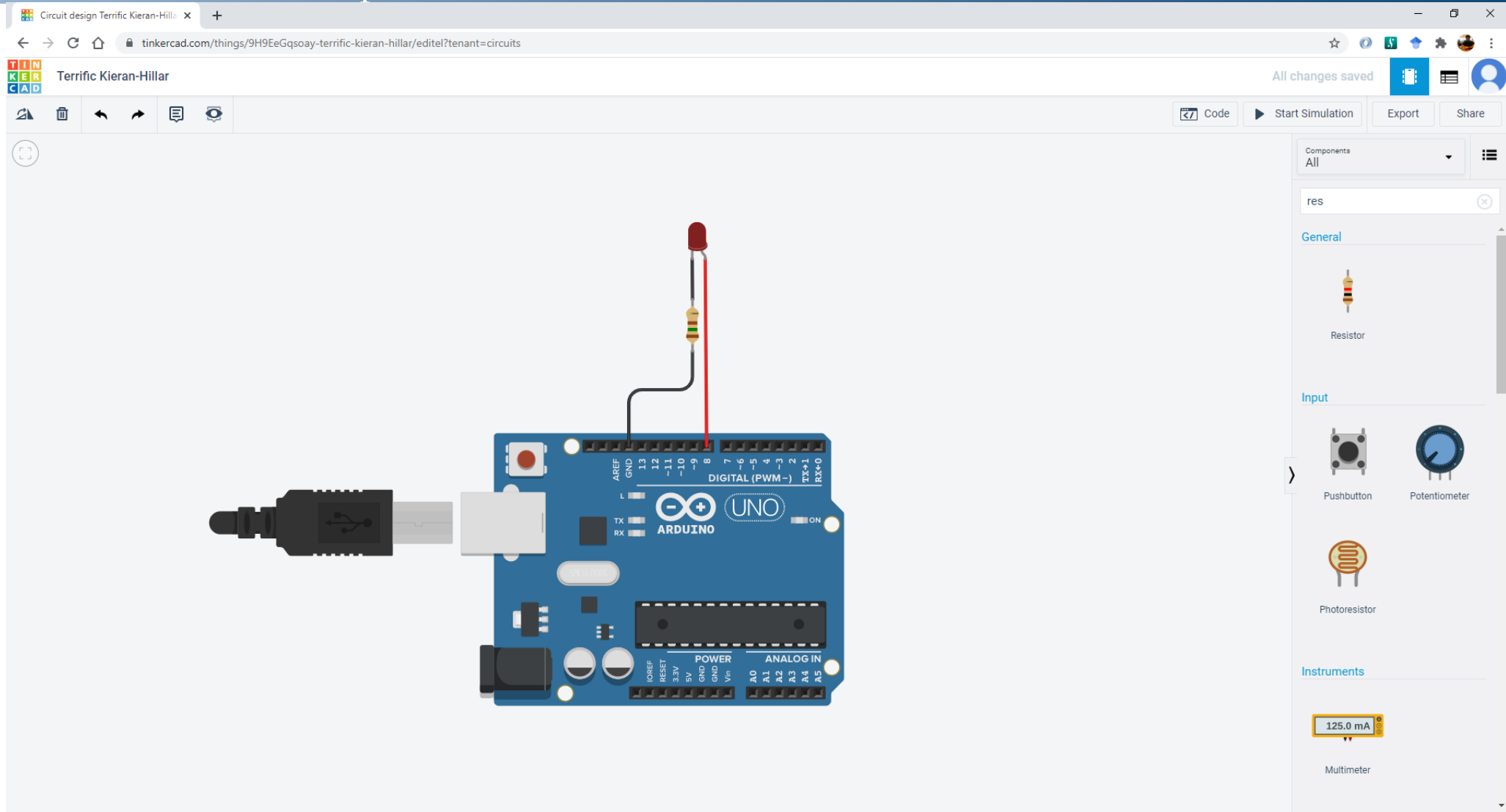
Resistor Component Properties:

Property	Value
Name	2
Resistance	150 Ω

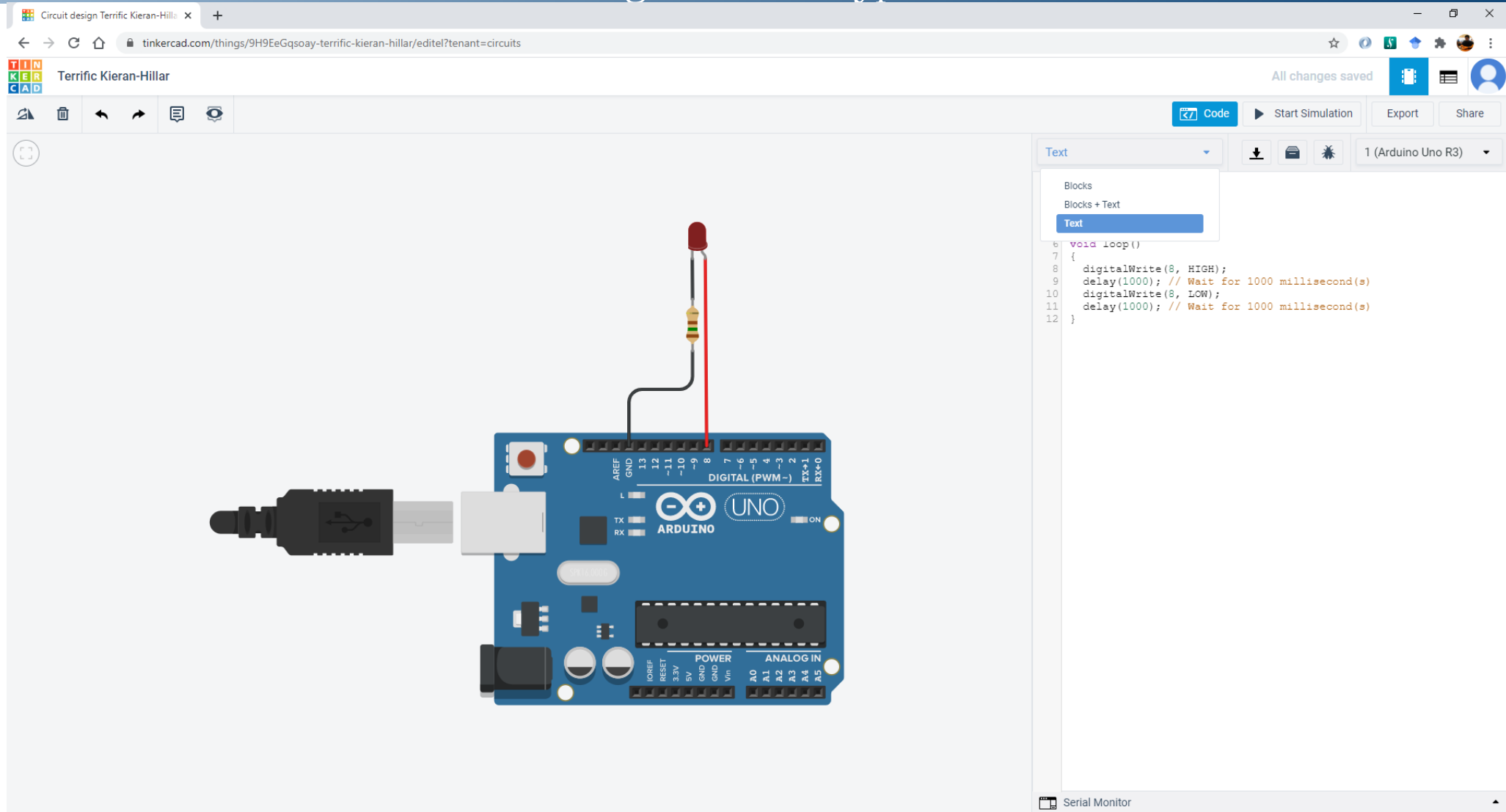
Component Selection Menu:

- Components: All
- Search: res
- General: Resistor
- Input: Pushbutton, Potentiometer
- Photoresistor
- Instruments: Multimeter (125.0 mA)

Tinkercad – Connect the components via cable



Tinkercad – Click the code button and change the code type Blocks to Text



The screenshot shows the Tinkercad web interface. In the center, an Arduino Uno R3 is connected to a red LED. The LED's anode is connected to digital pin 8, and its cathode is connected to ground. A USB cable is plugged into the Arduino's USB port.

On the right side, the code editor is open, and the 'Text' tab is selected. The code is as follows:

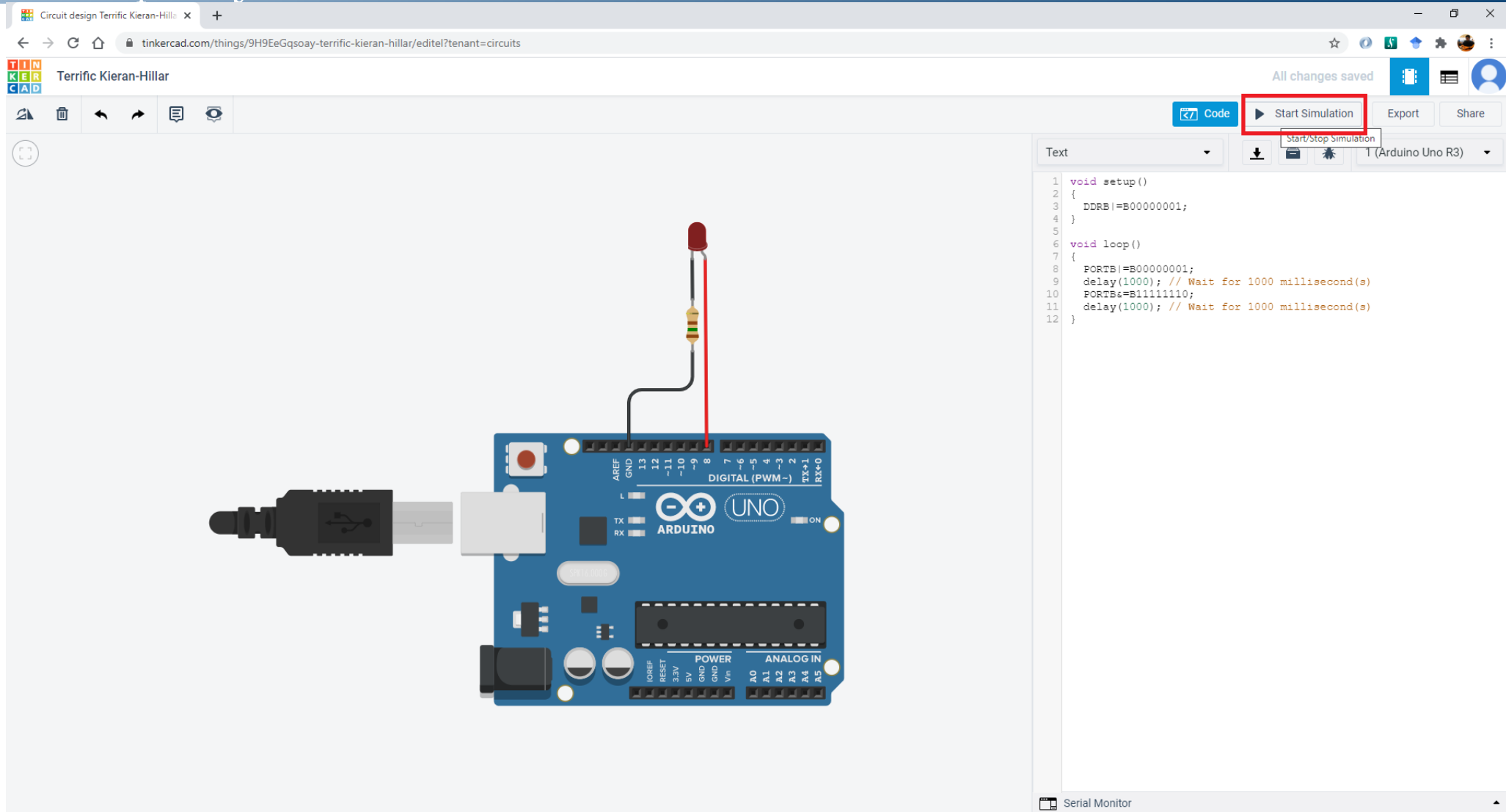
```

6 void loop()
7 {
8   digitalWrite(8, HIGH);
9   delay(1000); // Wait for 1000 millisecond(s)
10  digitalWrite(8, LOW);
11  delay(1000); // Wait for 1000 millisecond(s)
12 }

```

Below the code editor, the 'Serial Monitor' tab is visible but empty.

Tinkercad – Complete your code and start the simulation



The screenshot shows the Tinkercad web interface. In the center, an Arduino Uno R3 is connected to a red LED and a 220Ω resistor. The LED's anode is connected to digital pin 8, and the cathode is connected to ground. The resistor is connected in series with the LED. A USB Type-A cable is plugged into the USB port of the Arduino.

On the right side, the 'Code' tab is active. The 'Start Simulation' button is highlighted with a red box. Below it, the code editor shows the following code:

```

1 void setup()
2 {
3   DDRB |= B00000001;
4 }
5
6 void loop()
7 {
8   PORTB |= B00000001;
9   delay(1000); // Wait for 1000 millisecond(s)
10  PORTB |= B11111110;
11  delay(1000); // Wait for 1000 millisecond(s)
12 }
  
```

At the bottom of the interface, there is a 'Serial Monitor' tab.

Tinkercad – Debug your if there is a problem

Circuit design Terrific Kieran-Hillar

tinkercad.com/things/9H9EeGqsoay-terrific-kieran-hillar/editel?tenant=circuits

Terrific Kieran-Hillar

All changes saved

Code Start Simulation Export Share

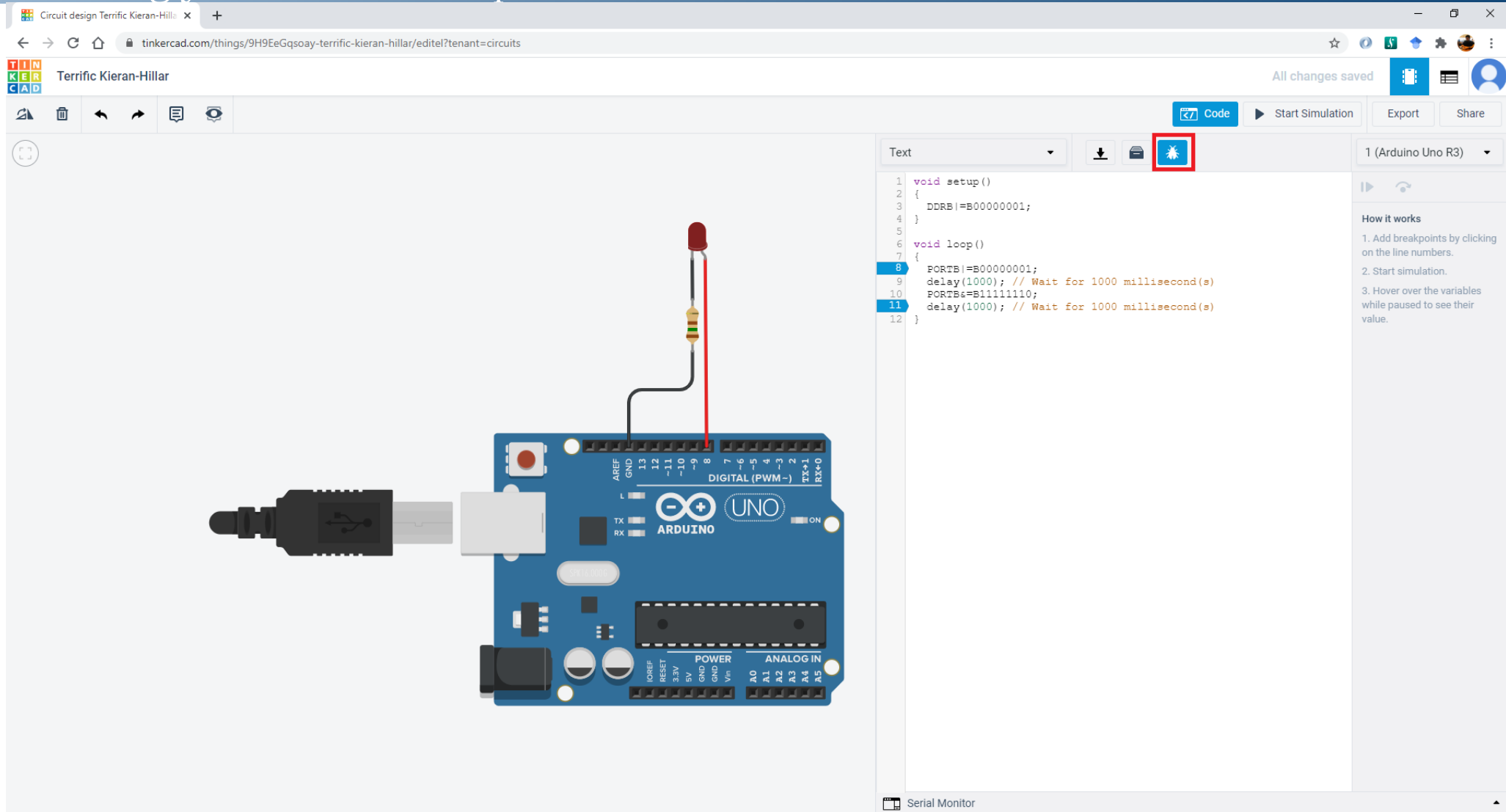
Text

1 void setup()
2 {
3 DDRB|=B00000001;
4 }
5
6 void loop()
7 {
8 PORTB|=B00000001;
9 delay(1000); // Wait for 1000 millisecond(s)
10 PORTB&=B11111110;
11 delay(1000); // Wait for 1000 millisecond(s)
12 }

1 (Arduino Uno R3)

How it works

1. Add breakpoints by clicking on the line numbers.
2. Start simulation.
3. Hover over the variables while paused to see their value.



Serial Monitor

Tinkercad – Use the serial monitor for communication between the microcontroller and the computer

Circuit design Terrific Kieran-Hillar

tinkercad.com/things/9H9EeGqsoay-terrific-kieran-hillar/editel?tenant=circuits

Terrific Kieran-Hillar

Simulator time: 00:01:09

All changes saved

Code Stop Simulation Export Share

Text 1 (Arduino Uno R3)

```

1 volatile uint8_t* port;
2
3 void setup()
4 {
5     DDRB |= B000000001;
6     port = &PORTB;
7     Serial.begin(115200);
8 }
9
10 void loop()
11 {
12     PORTB |= B000000001;
13     Serial.println(*port);
14     delay(1000); // Wait for 1000 millisecond(s)
15     PORTB |= B111111110;
16     Serial.println(*port);
17     delay(1000); // Wait for 1000 millisecond(s)
18 }

```

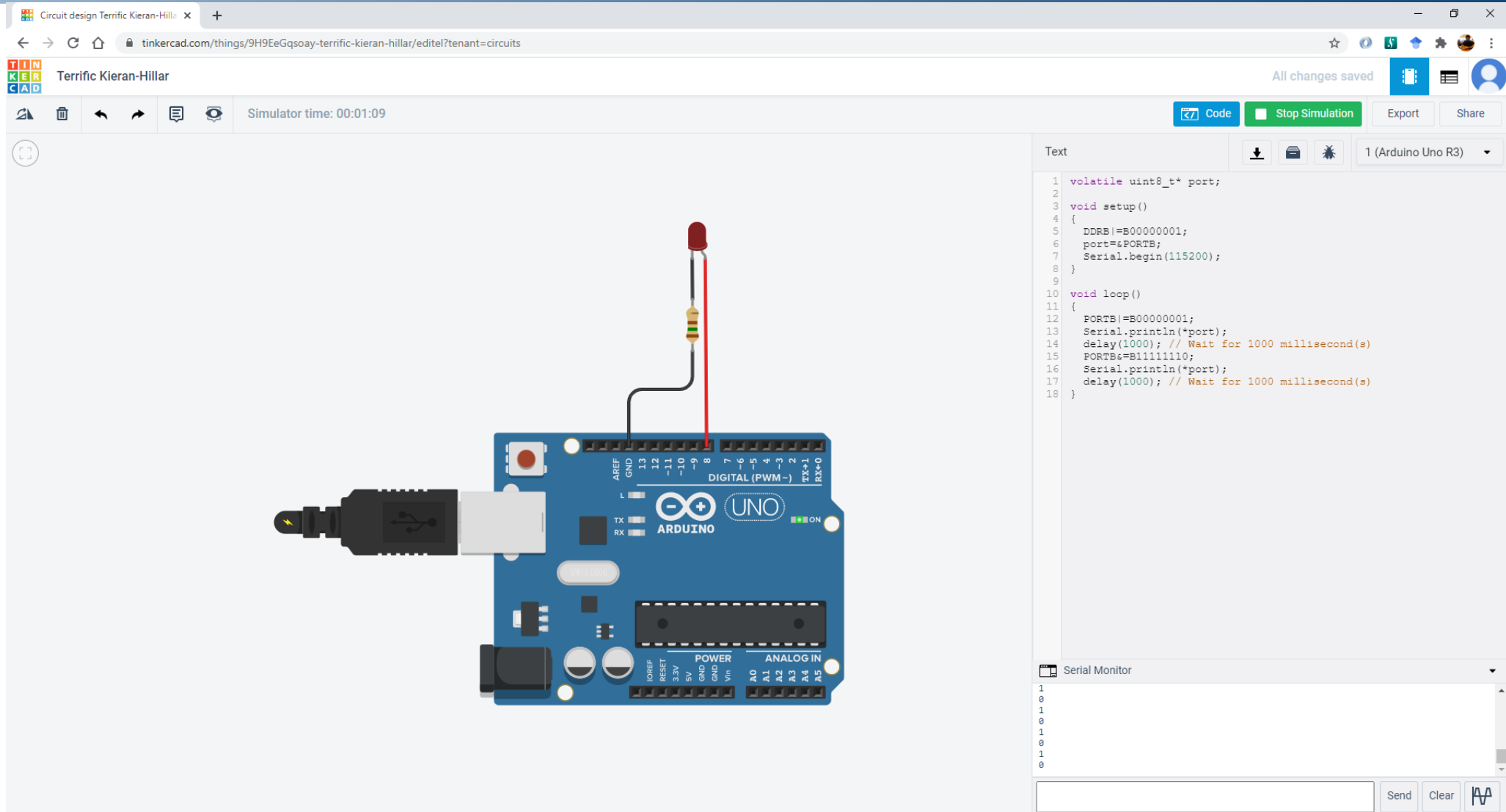
Serial Monitor

```

1
0
1
0
1
0
1
0

```

Send Clear





Thanks for your attention...

Questions, Comments ???