

Embedded Systems

Intro to AVR Microcontroller

Abdallah El Ghamry

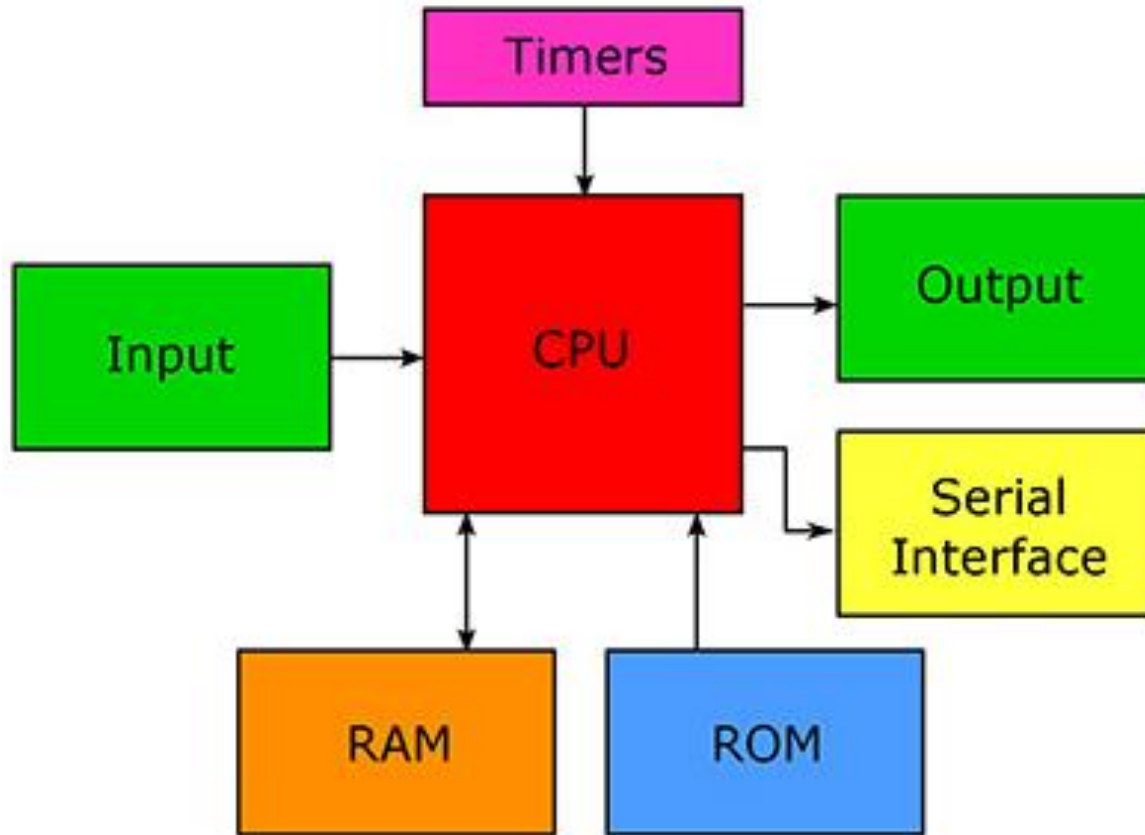


Intro to AVR Microcontroller

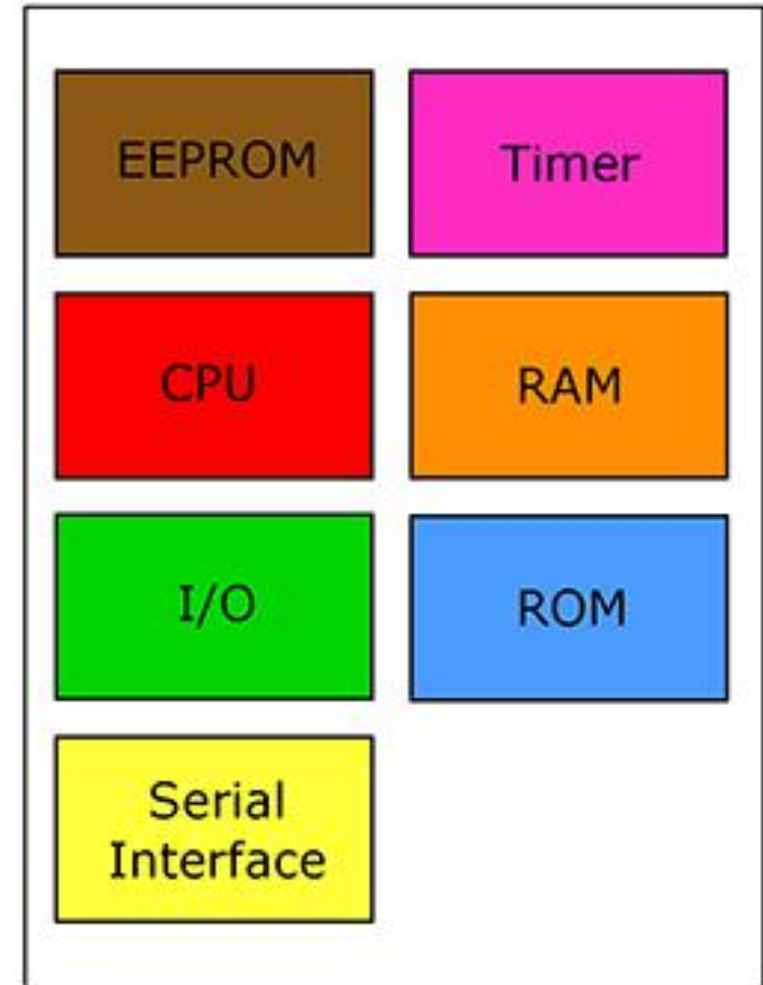
- The purpose of this Lab is to introduce a general knowledge about the **AVR microcontroller architecture**.
- As an example, we will study the famous **Atmega328P**, which is used in the **Arduino Uno board**.
- We will install **Atmel Studio** and know how to generate a **hex file**.
- Next, we will build our first **embedded C application**.

Microprocessor vs. Microcontroller

Microprocessor: CPU and several supporting chips.

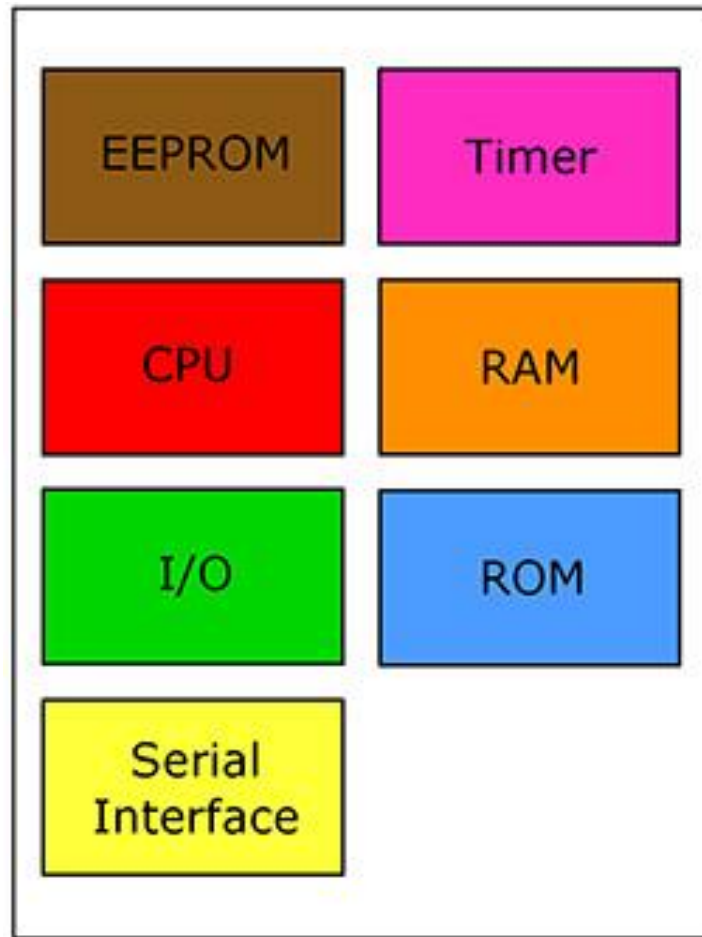


Microcontroller: CPU on a single chip.



Microcontroller

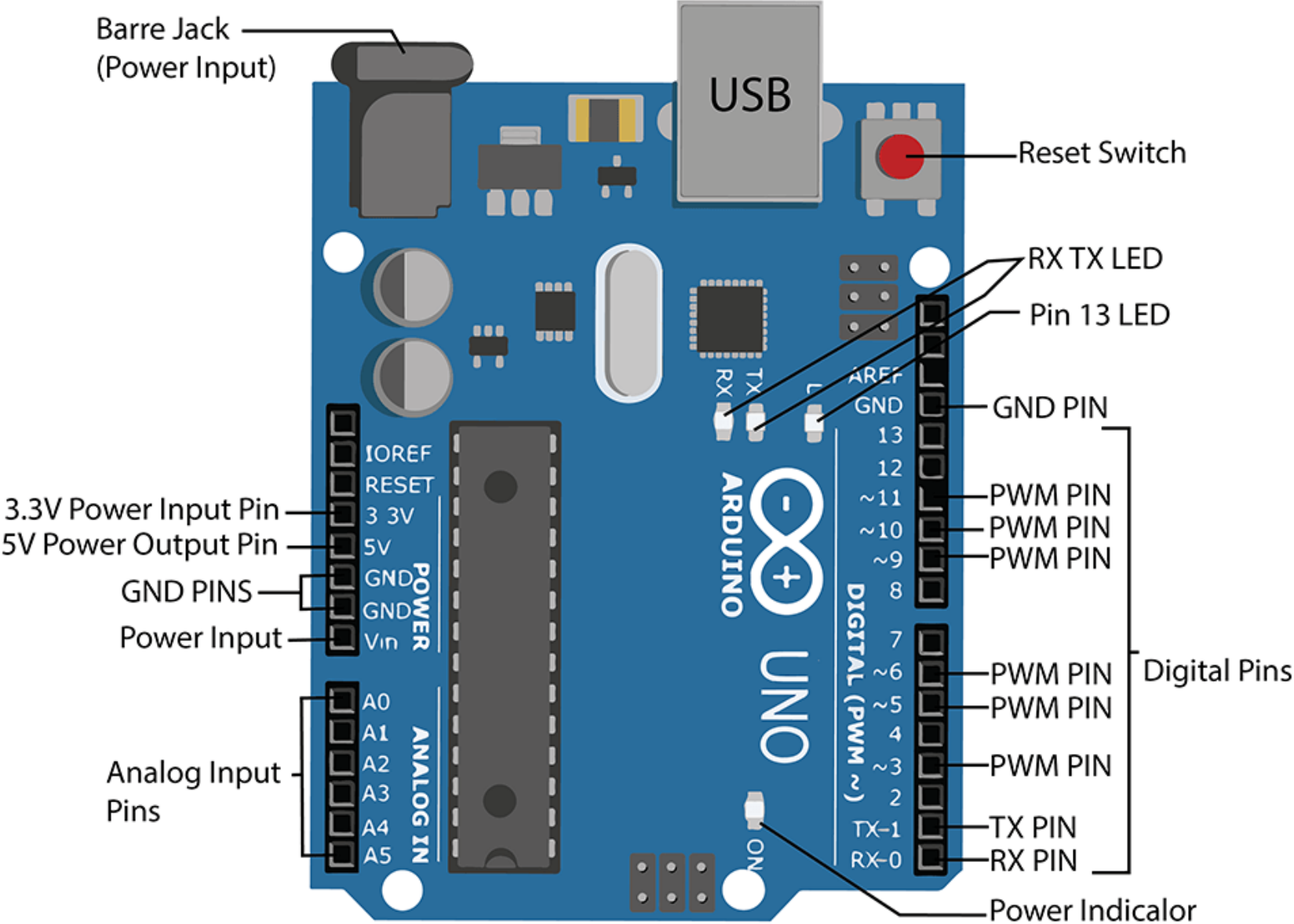
- A microcontroller is an integrated circuit consisting of a **complete computer on a single chip** and used for specified control functions.



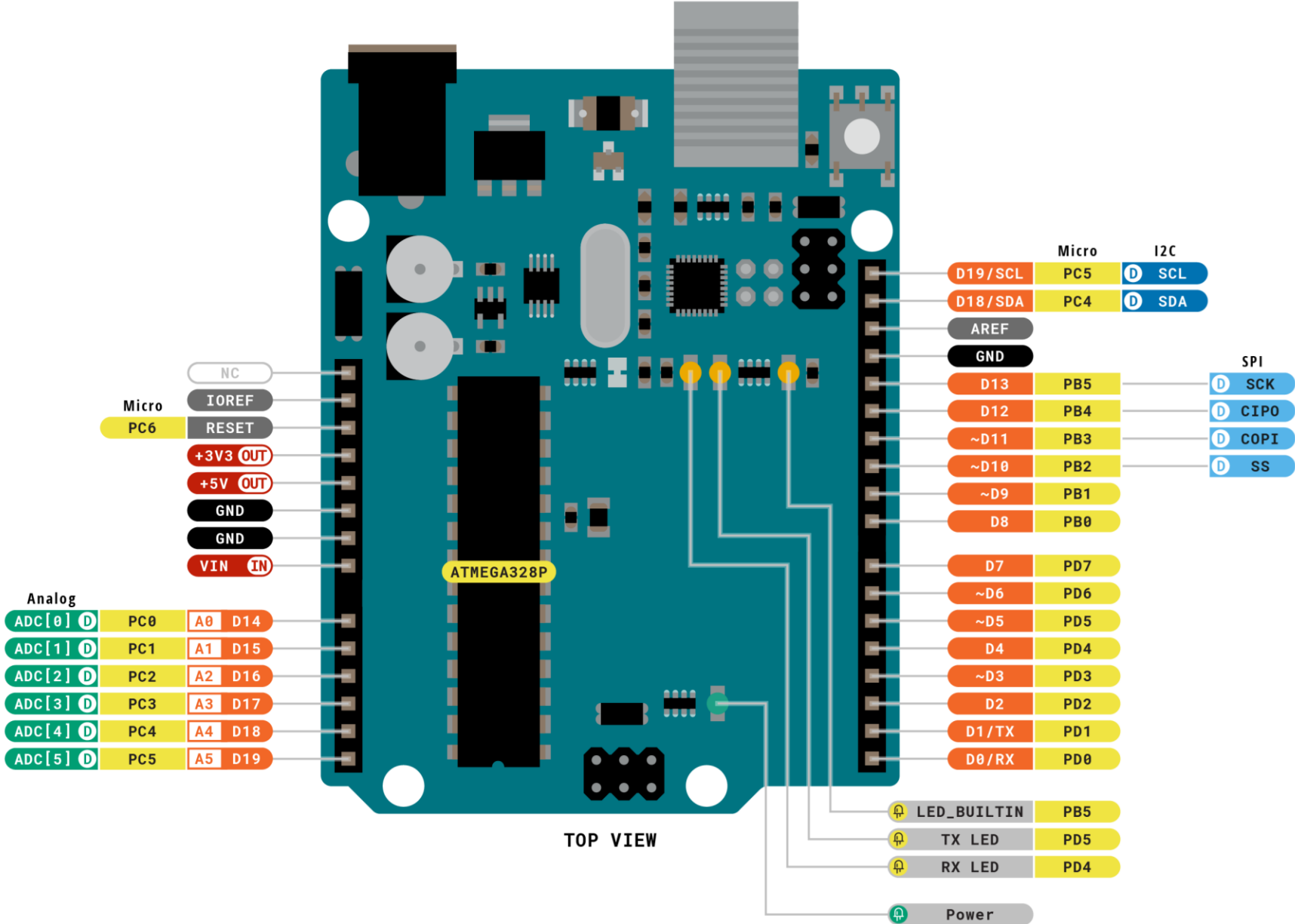
- Arduino is **open-source hardware** that can be used to develop **embedded systems** with **open-source software**.
- Arduino has gained **massive popularity** among **students** for making a working model.
- The reasons behind the popularity of Arduino are its **low cost**, **availability of software**, and **easy- to-interface** possibility.
- The Arduino environment has been designed to be **easy to use for beginners** who have **no software or electronics experience**.

- Arduino is used in many educational programs around the world, particularly by designers who want to easily create prototypes but do not need a deep understanding of the technical details.
- Because it is designed to be used by nontechnical people, the software includes plenty of example code to demonstrate how to use the Arduino board.
- People already working with microcontrollers are also attracted to Arduino because of its facility for quick implementation of ideas.

Arduino Uno: Board



Arduino Uno: Pinout Diagram



Arduino Uno: Tech Specs

Microcontroller	ATmega328P
Clock Speed	16 MHz
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Digital I/O Pins	14 (of which 6 PWM)
PWM Digital I/O Pins	6
Analog Input Pins	6
Operating Voltage	5V
DC Current per I/O Pin	20 mA

ATmega328P Microcontroller

- ATmega328P is a very advance and feature-rich microcontroller.
- It is one of the famous microcontrollers of Atmel because of its use in the Arduino UNO board.
- Created by Atmel, the ATmega328P is a single-chip microcontroller-based on an 8-bit RISC processor core.
- This small microcontroller is low-powered and affordable, making it an excellent choice for various applications.

ATmega328P Microcontroller: Datasheet



ATmega328P

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

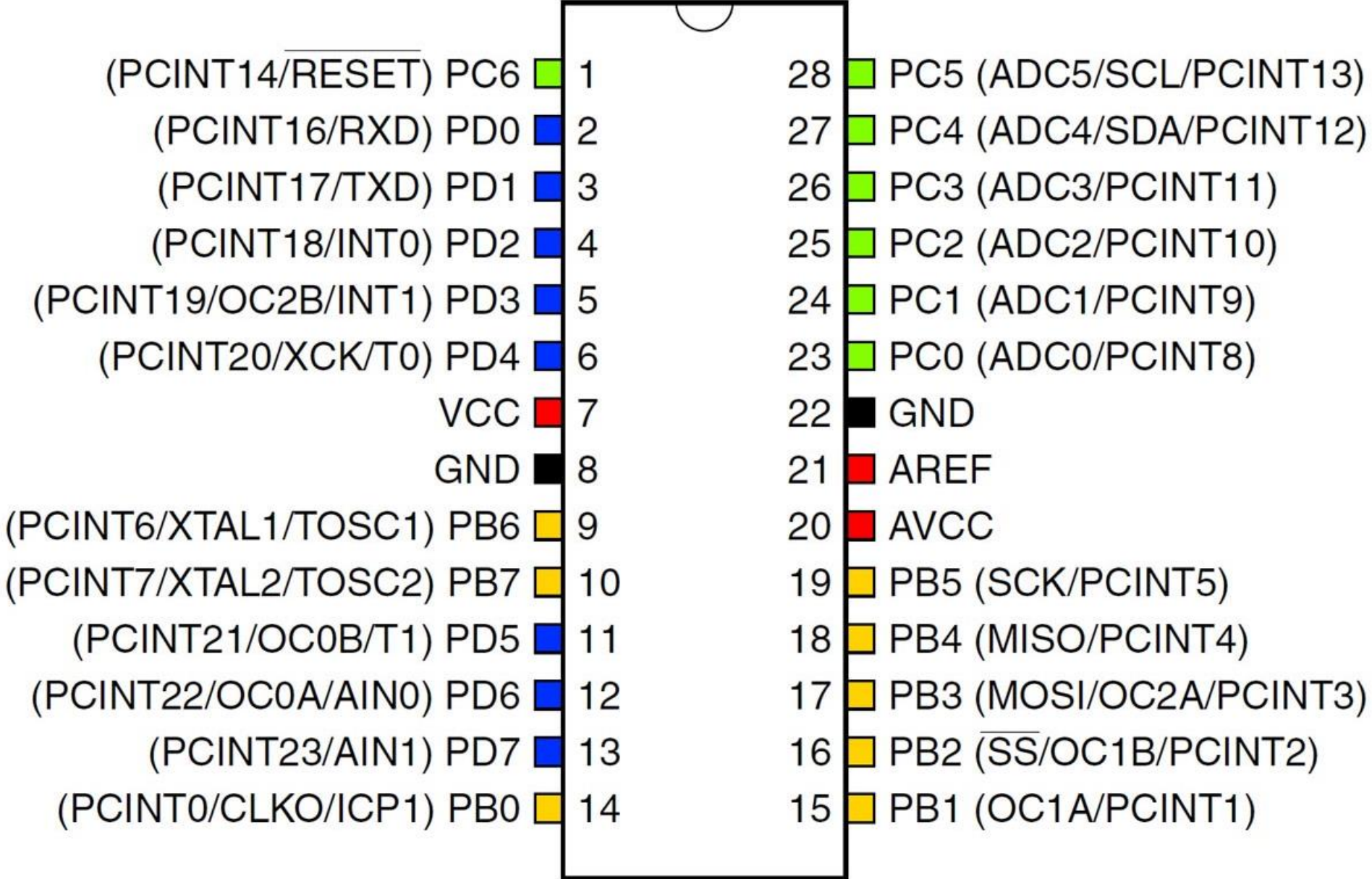
Features

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
 - 131 powerful instructions – most single clock cycle execution
 - 32 x 8 general purpose working registers
 - Fully static operation
 - Up to 16MIPS throughput at 16MHz
 - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
 - 32K bytes of in-system self-programmable flash program memory
 - 1Kbytes EEPROM
 - 2Kbytes internal SRAM
 - Write/erase cycles: 10,000 flash/100,000 EEPROM
 - Optional boot code section with independent lock bits
 - In-system programming by on-chip boot program
 - True read-while-write operation
 - Programming lock for software security
- Peripheral features
 - Two 8-bit Timer/Counters with separate prescaler and compare mode
 - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
 - Real time counter with separate oscillator
 - Six PWM channels
 - 8-channel 10-bit ADC in TQFP and QFN/MLF package
 - Temperature measurement
 - Programmable serial USART
 - Master/slave SPI serial interface
 - Byte-oriented 2-wire serial interface (Philips I²C compatible)
 - Programmable watchdog timer with separate on-chip oscillator
 - On-chip analog comparator
 - Interrupt and wake-up on pin change
- Special microcontroller features
 - Power-on reset and programmable brown-out detection
 - Internal calibrated oscillator
 - External and internal interrupt sources
 - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

7810D-AVR-01/15

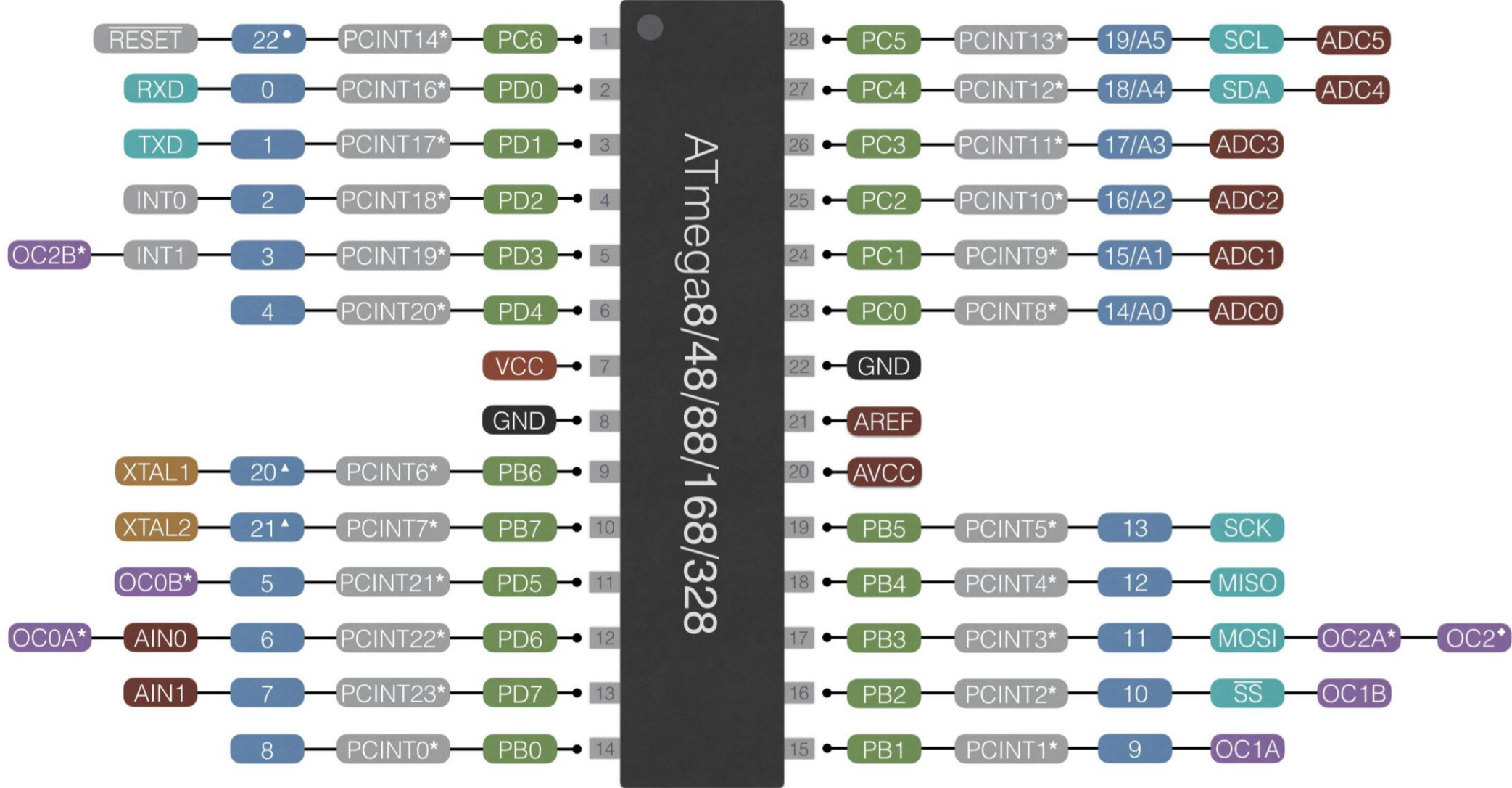
[ATmega328P Datasheet](#)

ATmega328P Microcontroller: Pinout



ATmega328P Microcontroller: Pinout

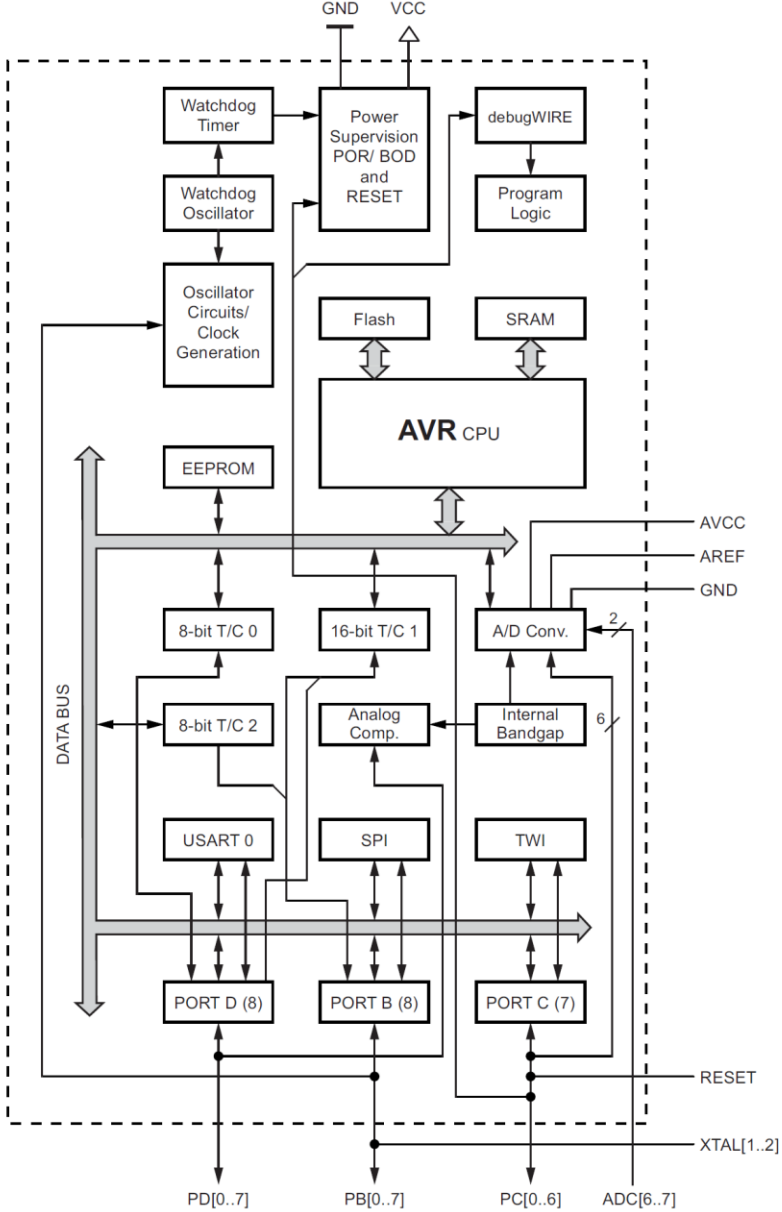
- POWER
- GROUND
- PORT PIN
- ARDUINO PIN
- ANALOG
- SERIAL INTERFACE
- TIMER / PWM PIN
- INTERRUPT



ATmega328P Microcontroller: GPIO

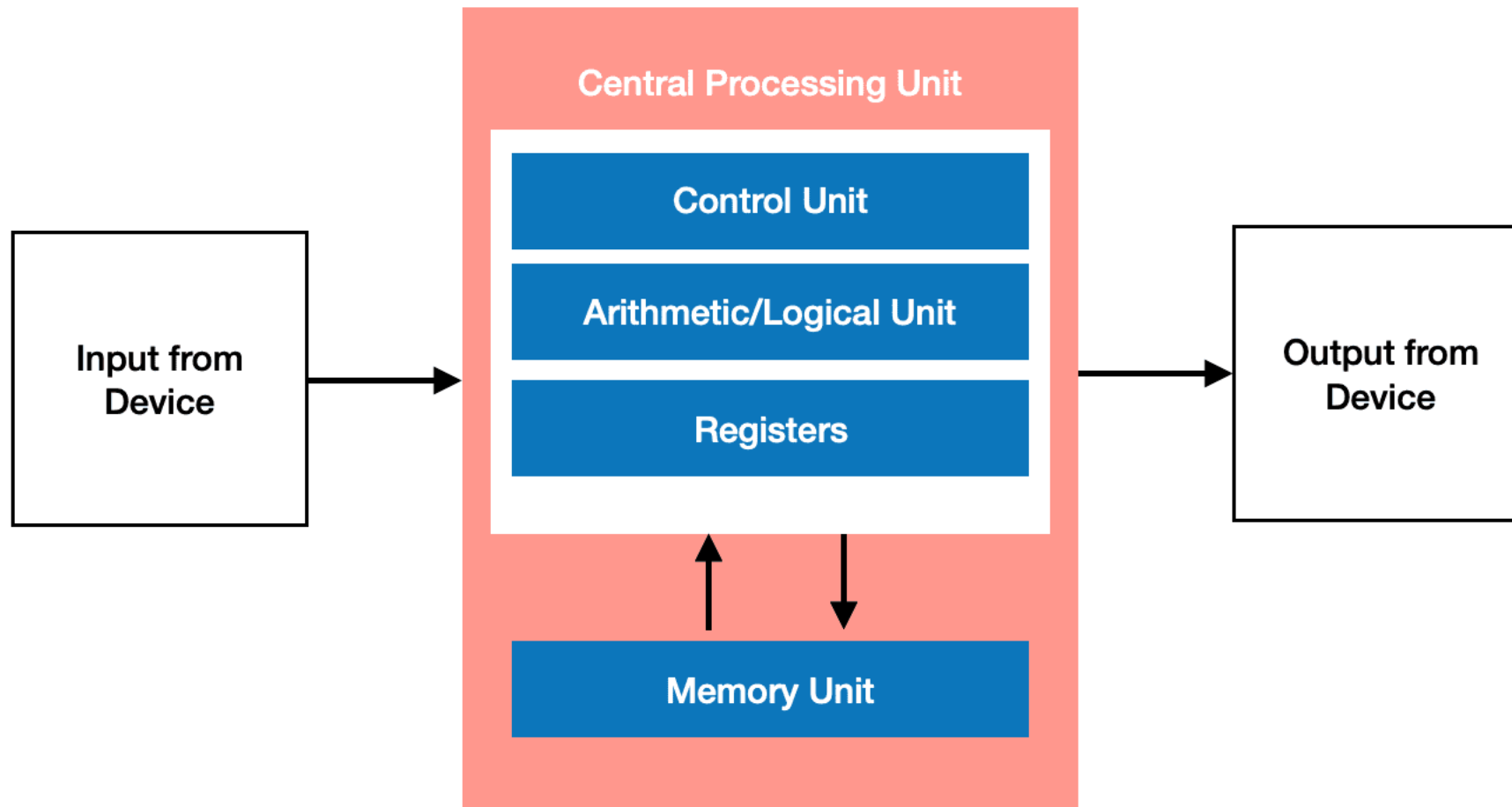
- The **GPIO** (General Purpose Input/Output) pins is a **software-controlled interface** that is usually found on microcontrollers.
- The GPIO pins **have no special purpose** in themselves.
- A GPIO pin is a **generic pin** whose behavior can be **programmed through software**.
- It facilitates an **optional ability** for device designers to create an **interface/connection** between the IC and a **peripheral component** by programming some hardware registers.
- A GPIO pin can be either **Input** or **Output**.

ATmega328P Microcontroller: Block Diagram



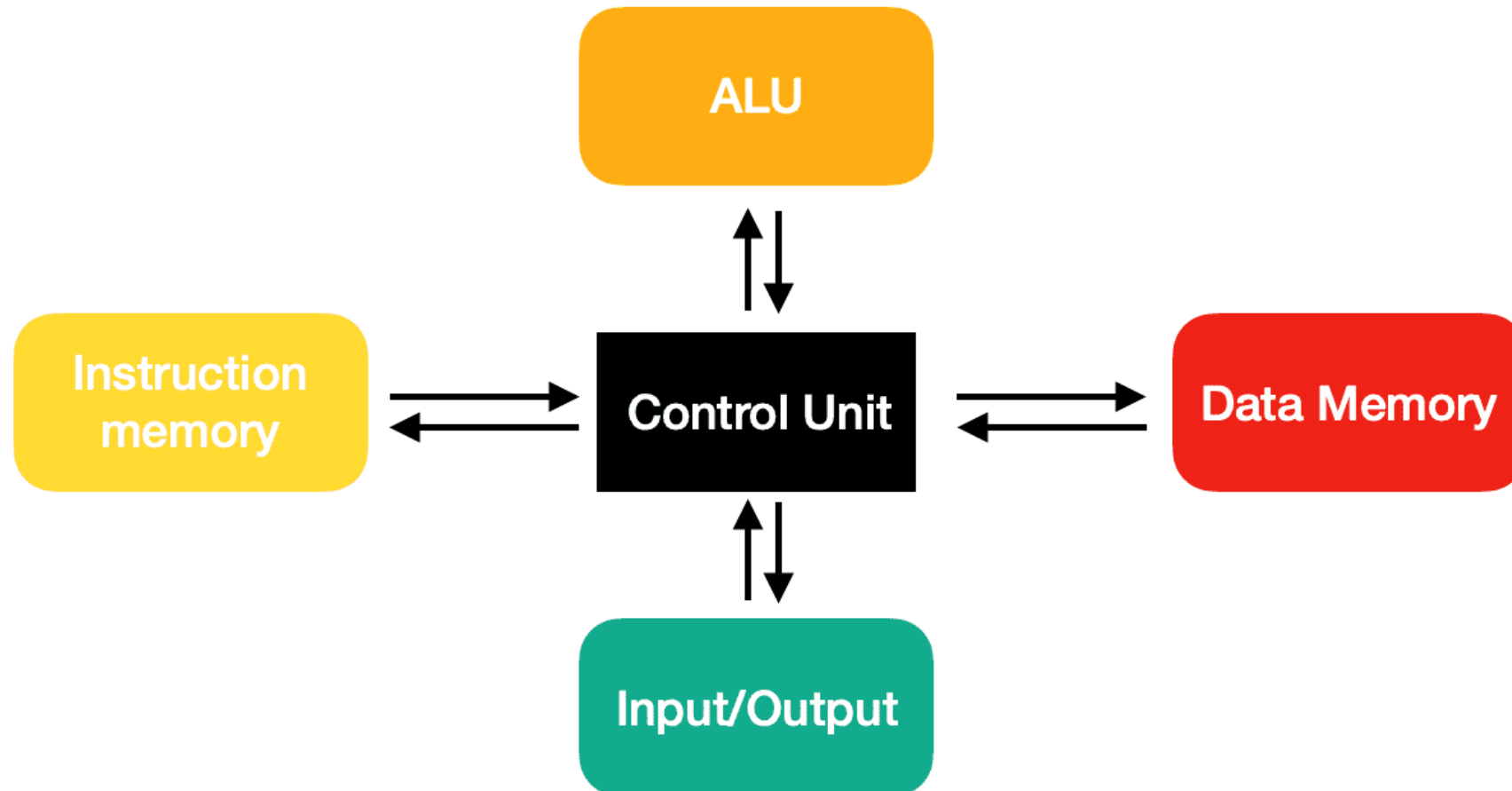
Von Neumann Architecture

- The **Von Neumann architecture** is based on the fact that the **program data and the instruction data are stored in the same memory unit.**



Harvard Architecture

- The **Harvard architecture** stores the **data** and **instructions separately**, therefore splitting the memory unit.



ATmega328P Microcontroller: Block Diagram

- In order to maximize performance and parallelism, the AVR uses a **Harvard architecture** – with separate memories and buses for program and data.

Flash
Program
Memory

Data
SRAM

ATmega328P Microcontroller: Ports

	7	6	5	4	3	2	1	0
PORTB	PB7	PB6	PB5	PB4	PB2	PB2	PB1	PB0

	7	6	5	4	3	2	1	0
PORTC	-	PC6	PC5	PC4	PC2	PC2	PC1	PC0

	7	6	5	4	3	2	1	0
PORTD	PD7	PD6	PD5	PD4	PD2	PD2	PD1	PD0

ATmega328P Microcontroller: Port B

7	6	5	4	3	2	1	0
PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

Port B Data Register

7	6	5	4	3	2	1	0
DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0

Port B Data Direction Register (DDRB)

ATmega328P Microcontroller: Port C

7	6	5	4	3	2	1	0
-	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0

Port C Data Register

7	6	5	4	3	2	1	0
-	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0

Port C Data Direction Register (DDRC)

ATmega328P Microcontroller: Port D

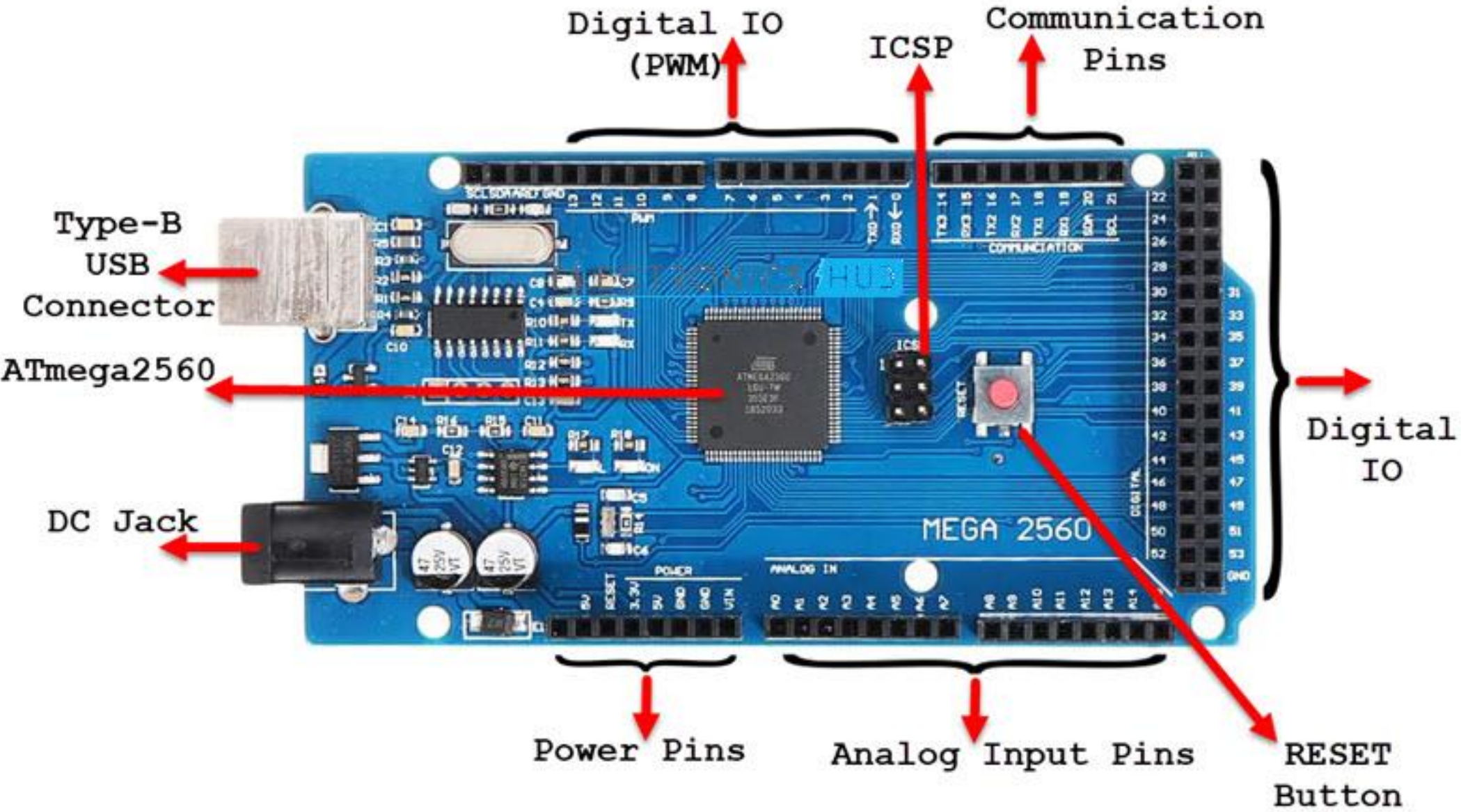
7	6	5	4	3	2	1	0
PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0

Port D Data Register

7	6	5	4	3	2	1	0
DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0

Port D Data Direction Register (DDRD)

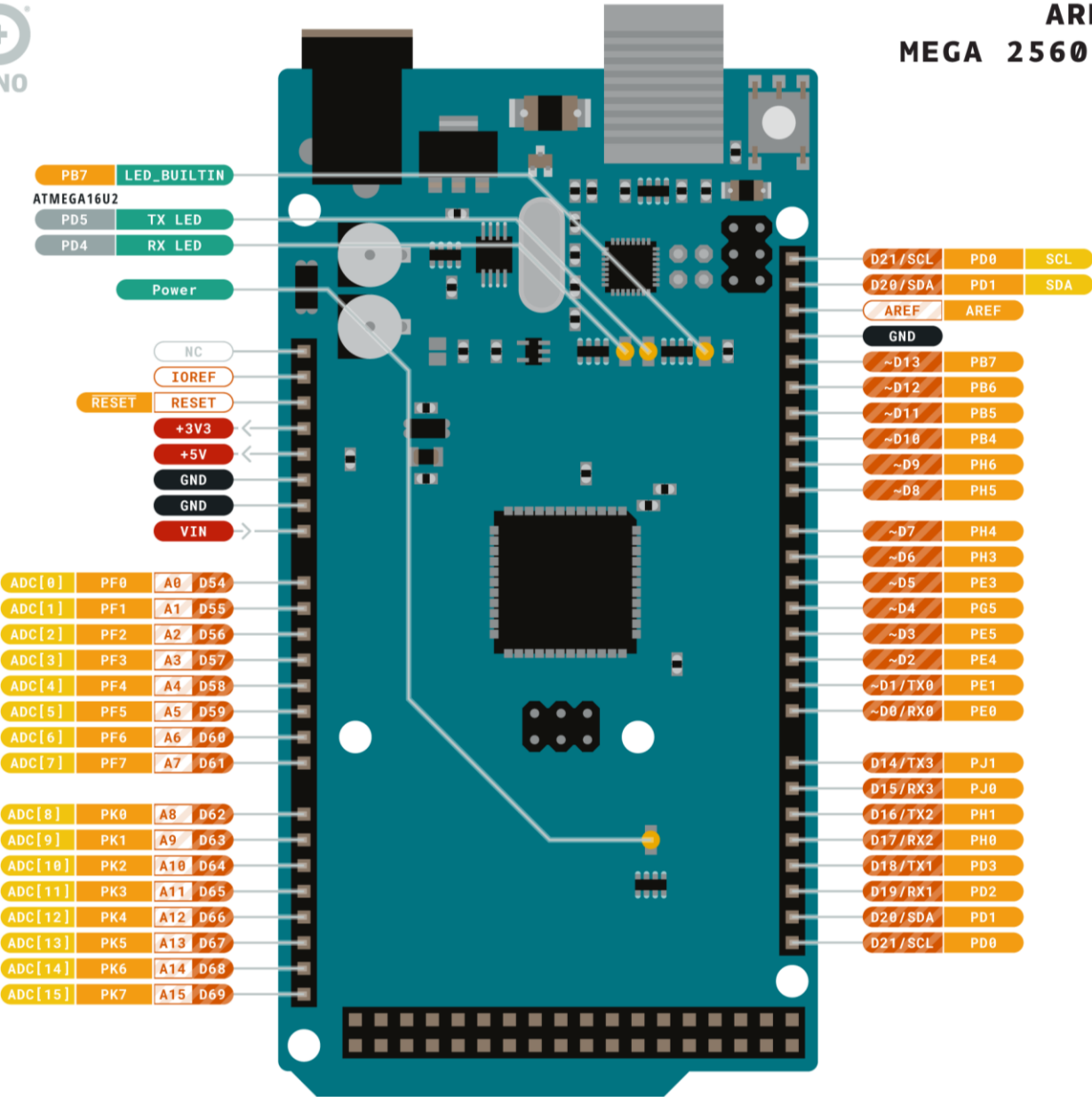
Arduino Mega: Board



Arduino Mega: Pinout Diagram



**ARDUINO
MEGA 2560 REV3**



Arduino Mega: Tech Specs

Microcontroller	ATmega2560
Clock Speed	16 MHz
Flash Memory	256 KB
SRAM	8 KB
EEPROM	4 KB
Digital I/O Pins	54 (of which 15 PWM)
PWM Digital I/O Pins	15
Analog Input Pins	16
Operating Voltage	5V
DC Current per I/O Pin	20 mA

ATmega2560 Microcontroller: Datasheet



Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

DATASHEET

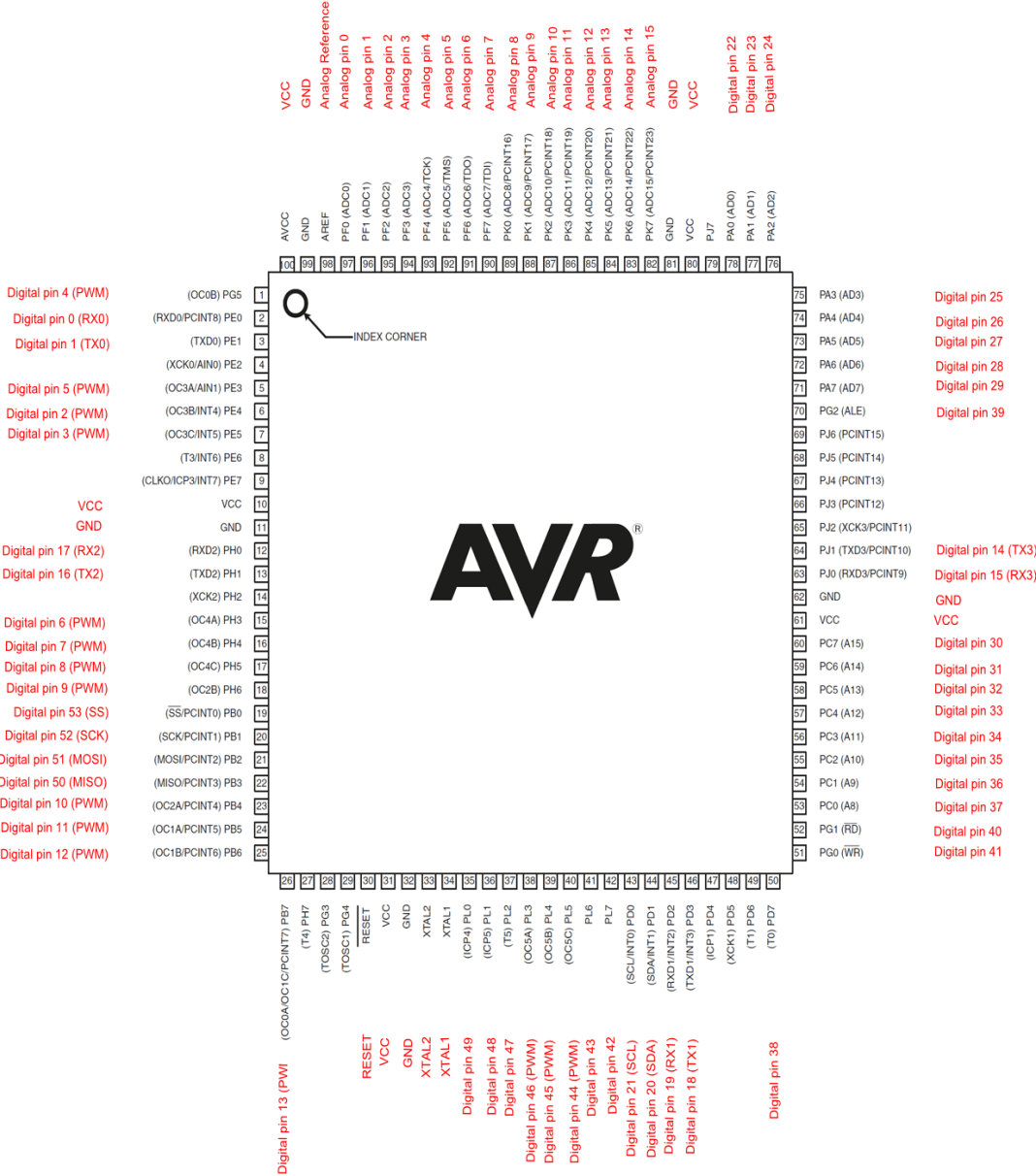
Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 135 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16MHz
 - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 64K/128K/256K Bytes of In-System Self-Programmable Flash
 - 4K Bytes EEPROM
 - 8K Bytes Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data Retention: 20 years at 85°C/100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - Programming Lock for Software Security
- Atmel® QTouch® Entry support
 - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix acquisition
 - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - Four 16-bit Timer/Counter with Separate Prescaler, Compare, and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Four 8-bit PWM Channels
 - Six/Ten PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
 - Output Compare Modulator
 - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
 - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
 - Master/Slave SPI Serial Interface
 - Byte Oriented 2-wire Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
 - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 54/96 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
 - 64-pad QFN/J, 64-lead TQFP (ATmega1281/2561)
 - 105-lead TQFP, 105-ball BGA (ATmega640/1280/2560)
 - RoHS/Full Green
- Temperature Range:
 - 40°C to 85°C Industrial
- Ultra-Low Power Consumption
 - Active Mode: 1MHz, 1.8V: 500µA
 - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
 - ATmega40V/ATmega1280V/ATmega1281V:
 - 0 - 8MHz @ 1.8V - 5.5V @ 0.8MHz @ 2.7V - 5.5V
 - ATmega20V/ATmega64V:
 - 0 - 20MHz @ 1.8V - 5.5V @ 0.8MHz @ 2.7V - 5.5V
 - ATmega40V/ATmega1280/ATmega1281:
 - 0 - 8MHz @ 2.7V - 5.5V @ 0.8MHz @ 4.5V - 5.5V
 - ATmega20V/ATmega64V:
 - 0 - 16MHz @ 4.5V - 5.5V

2549Q AVR-02/2014

[ATmega2560 Datasheet](#)

ATmega2560 Microcontroller: Pinout




- Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging **ARM-based** and **AVR microcontroller** (MCU) applications.
- Studio 7 supports all **AVR** and SMART MCUs.
- The Atmel Studio 7 IDP gives you an **easy-to-use environment** to **write, build** and **debug** your apps written in **C/C++** or **assembly**.

Atmel Studio 7: Setup Steps

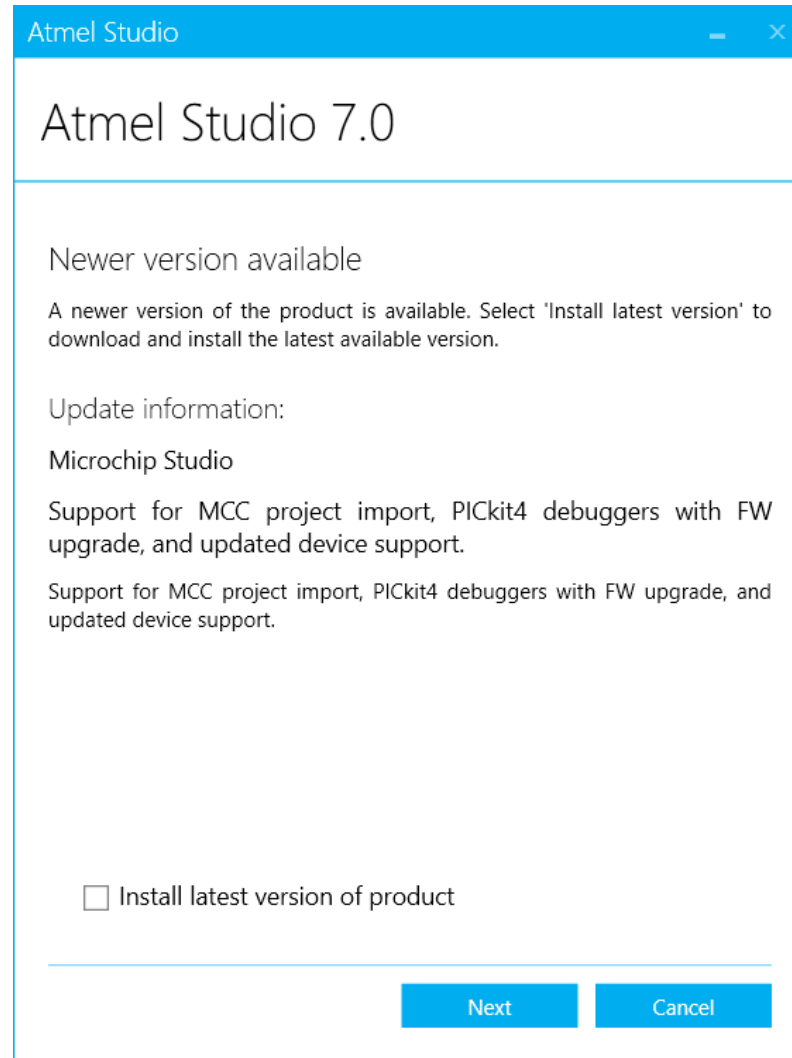
1. Download **Atmel Studio v7.0.1931** from [microchip](#).

Atmel Studio 7 IDE Archives

Web Installer (recommended)	Offline Installer
Atmel Studio v7.0.2389	Atmel Studio v7.0.2389
Atmel Studio v7.0.1931	Atmel Studio v7.0.1931 
Atmel Studio v7.0.1645	Atmel Studio v7.0.1645

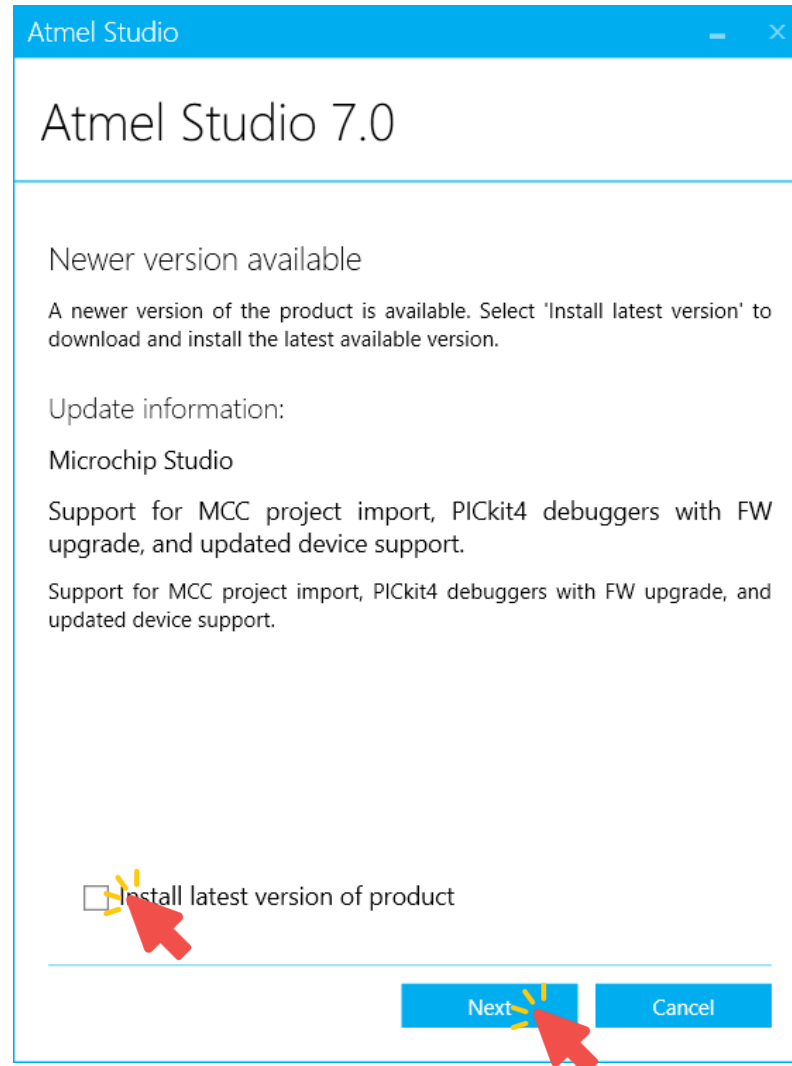
Atmel Studio 7: Setup Steps

2. Double click on the downloaded file to start installation.



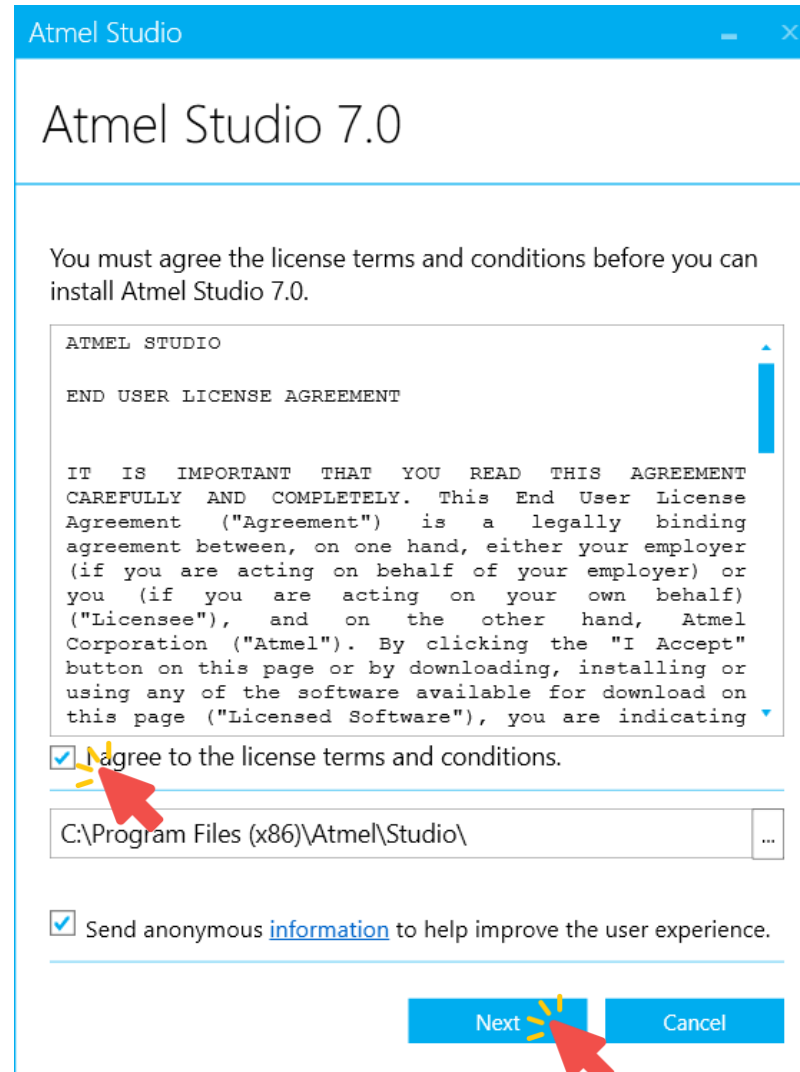
Atmel Studio 7: Setup Steps

3. Uncheck Install latest version of product, and click Next.



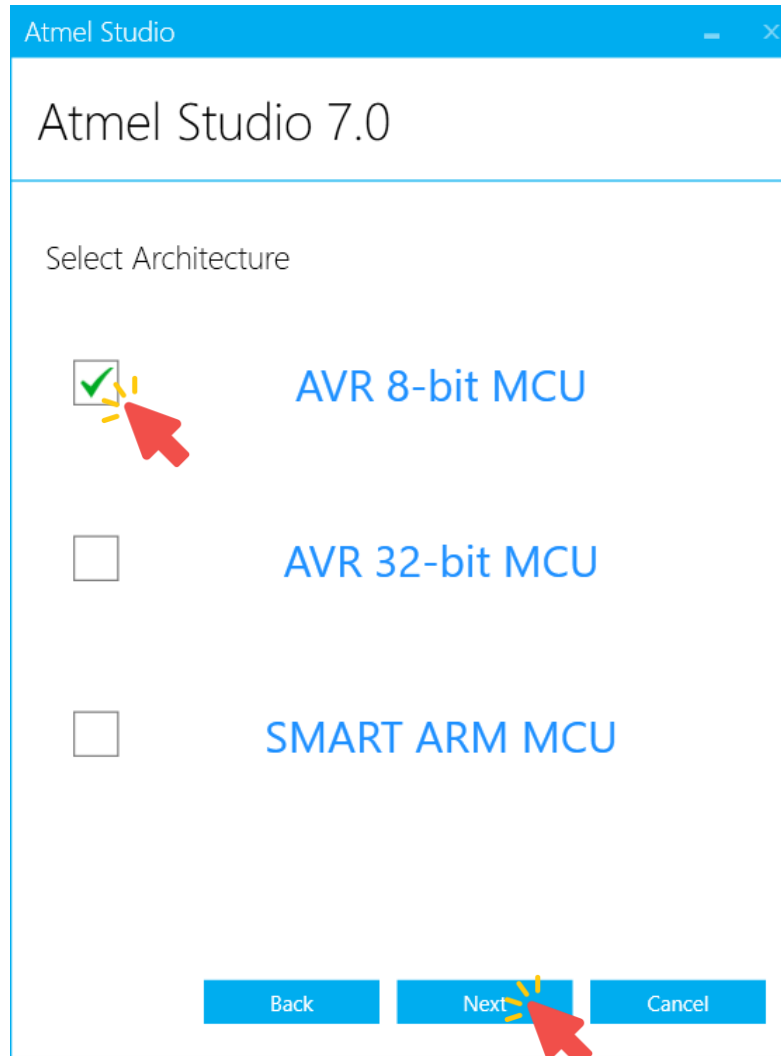
Atmel Studio 7: Setup Steps

4. Check I agree to the license terms and conditions, and click Next.



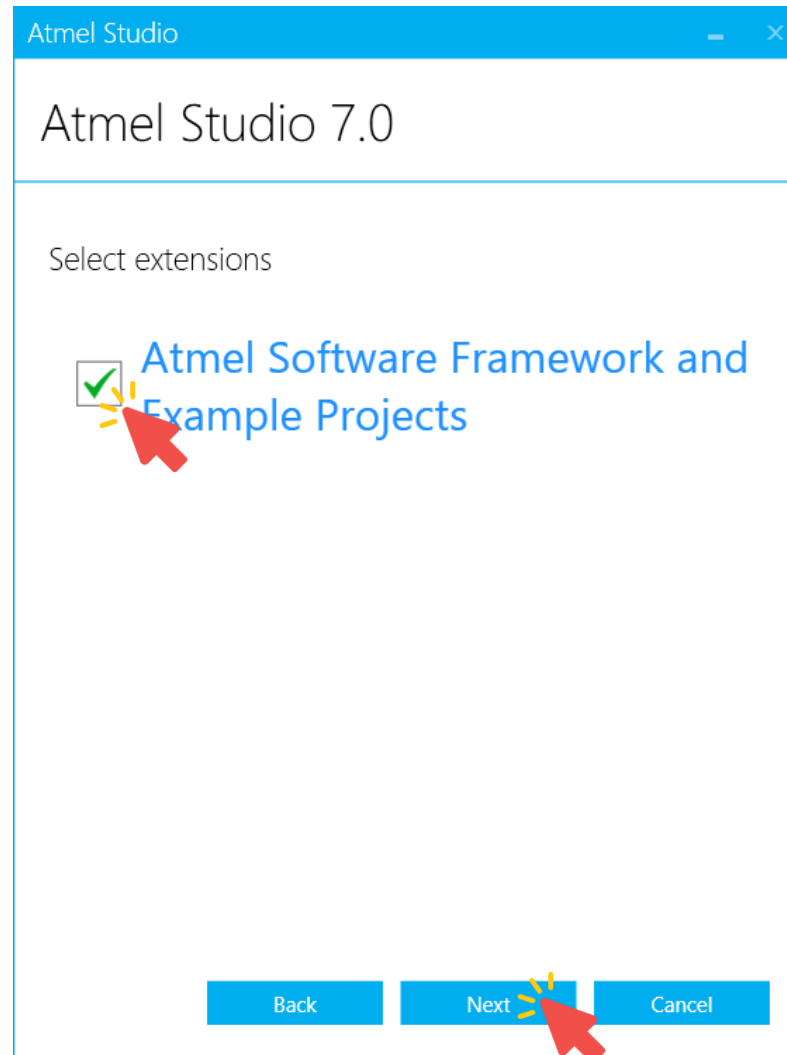
Atmel Studio 7: Setup Steps

5. Check *AVR 8-bit MCU*, and click *Next*.



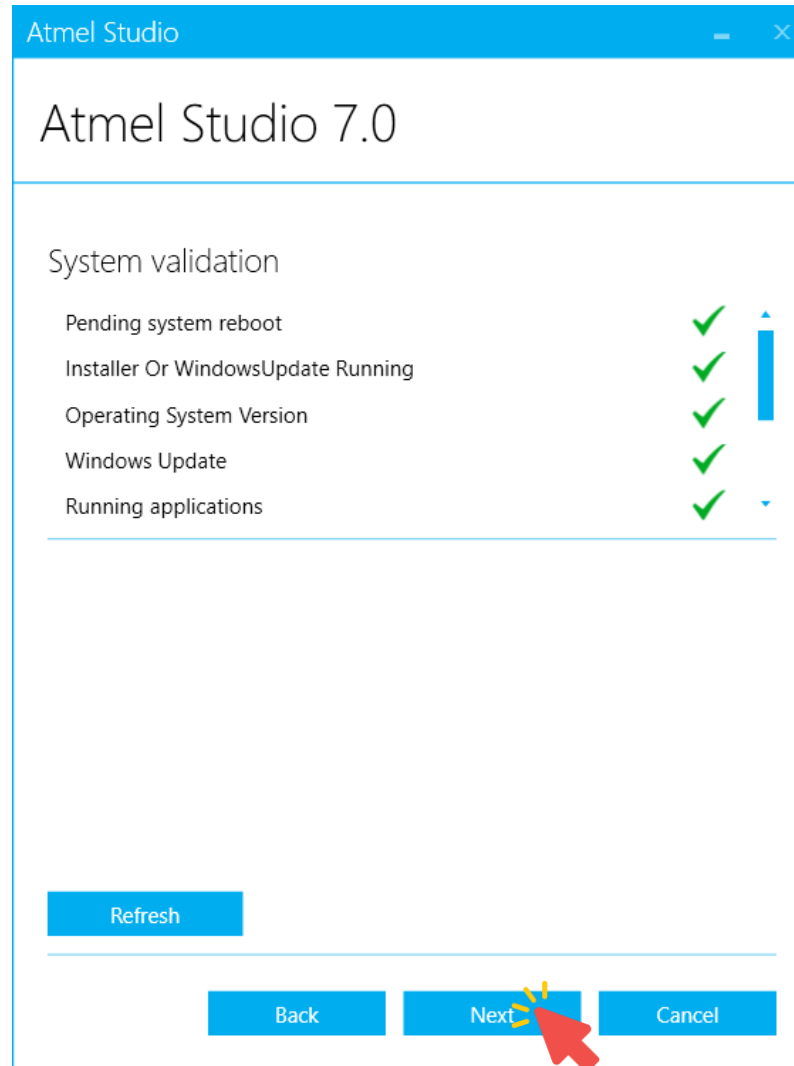
Atmel Studio 7: Setup Steps

6. Check **Atmel Software Framework and Examples**, and click **Next**.



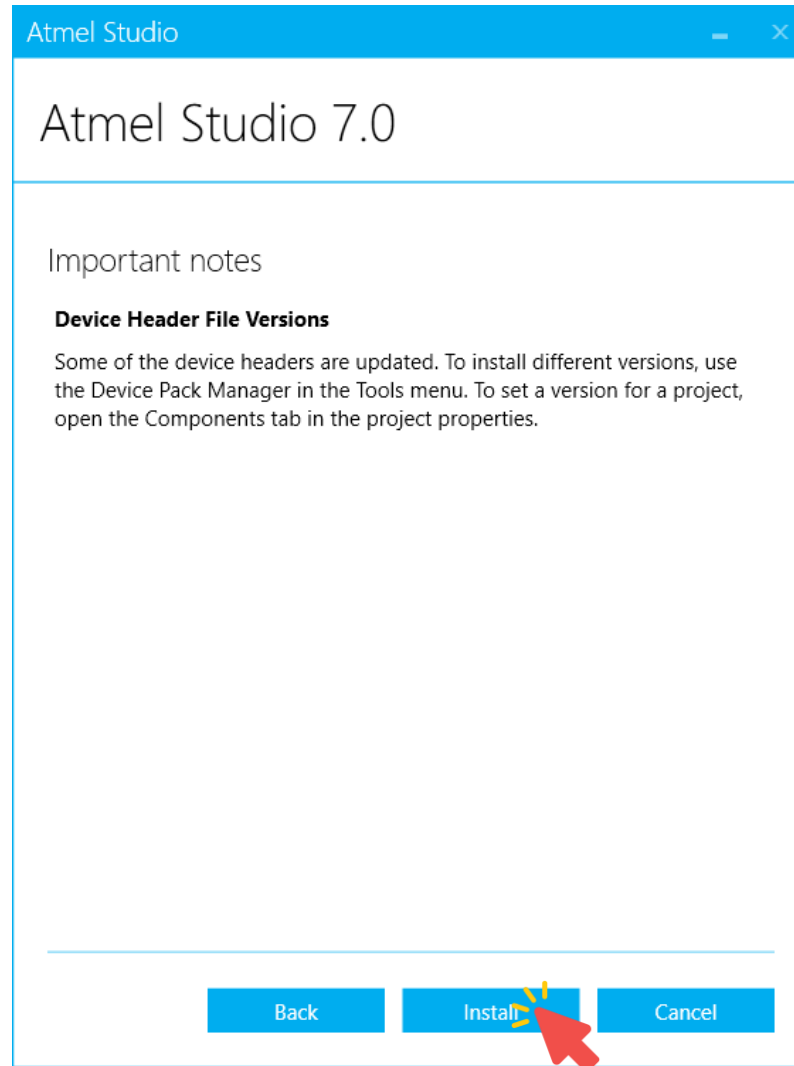
Atmel Studio 7: Setup Steps

7. Click Next.



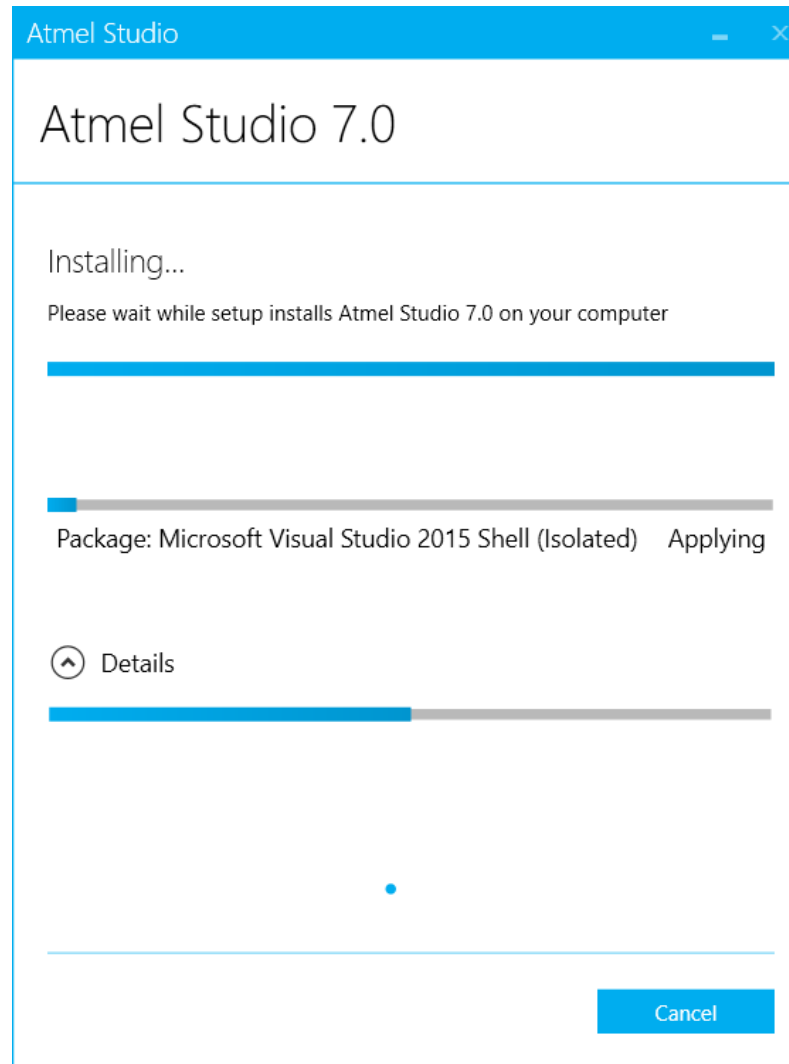
Atmel Studio 7: Setup Steps

8. Click Install.



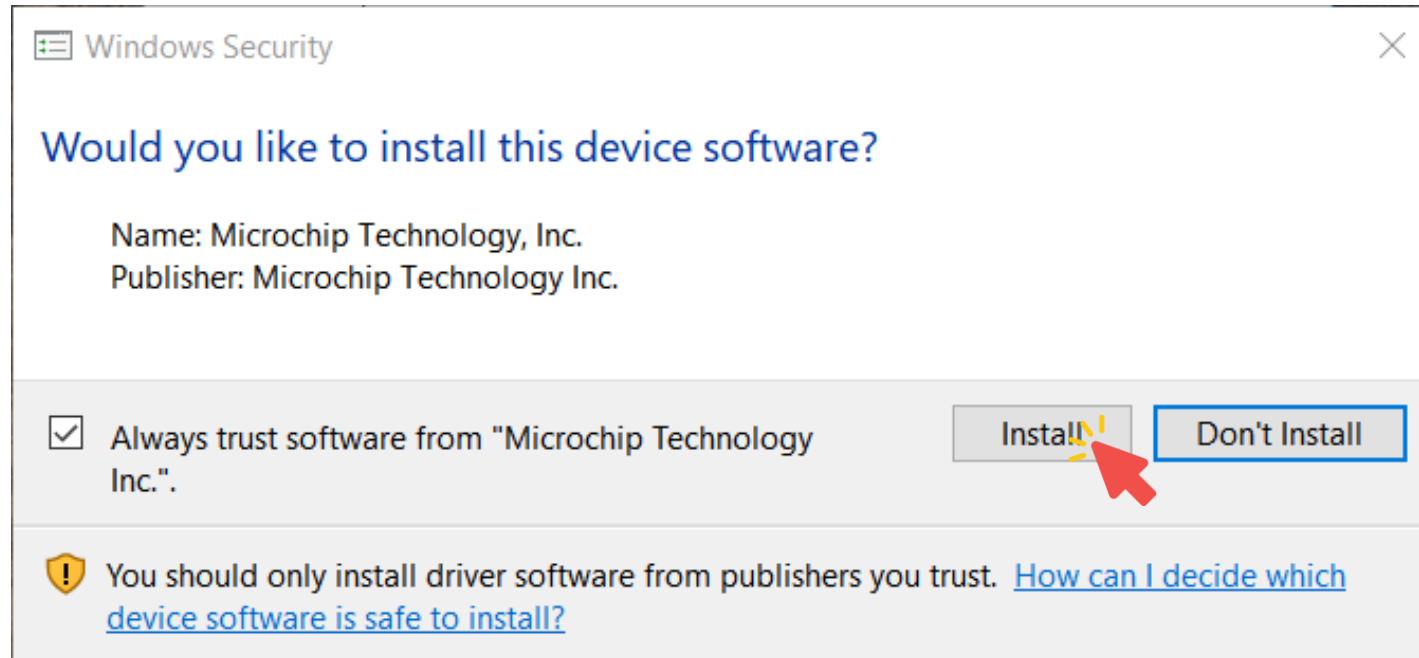
Atmel Studio 7: Setup Steps

9. Wait until setup is finished. This will take **some time**.



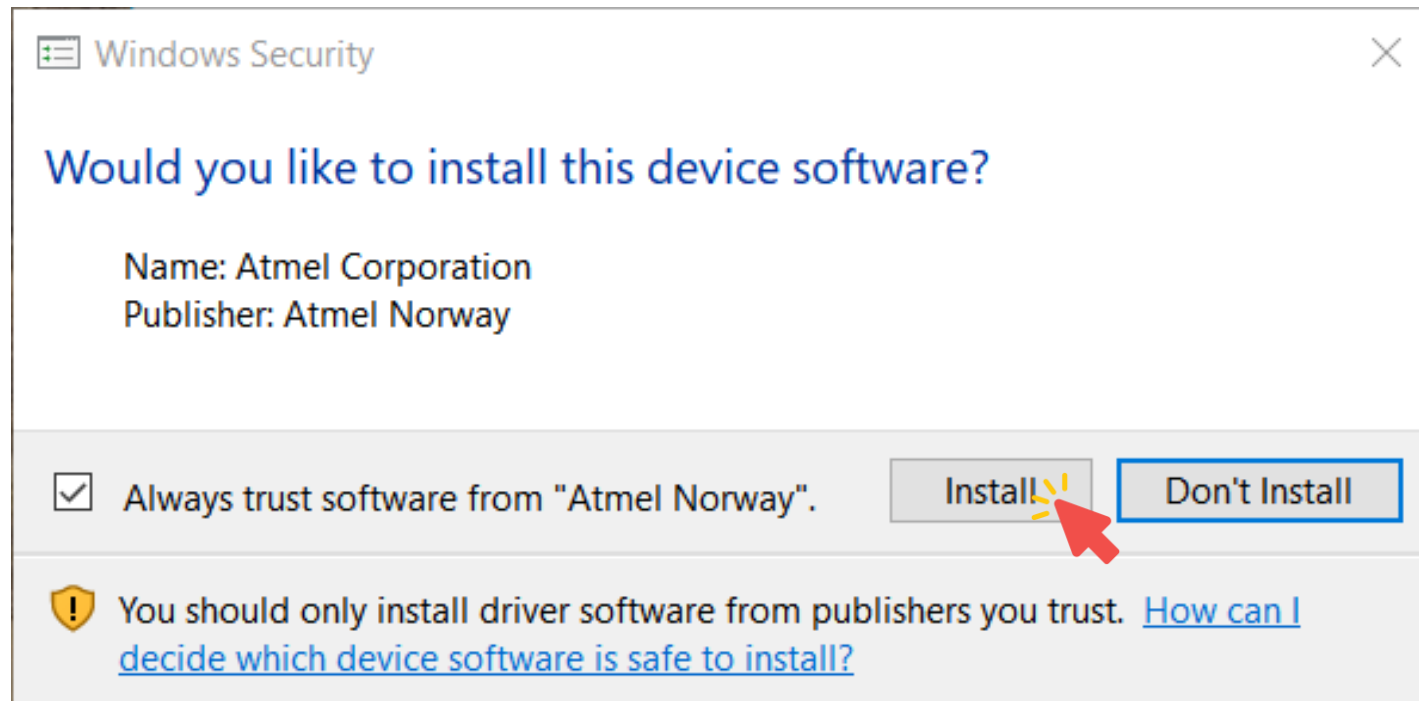
Atmel Studio 7: Setup Steps

10. Click Install.



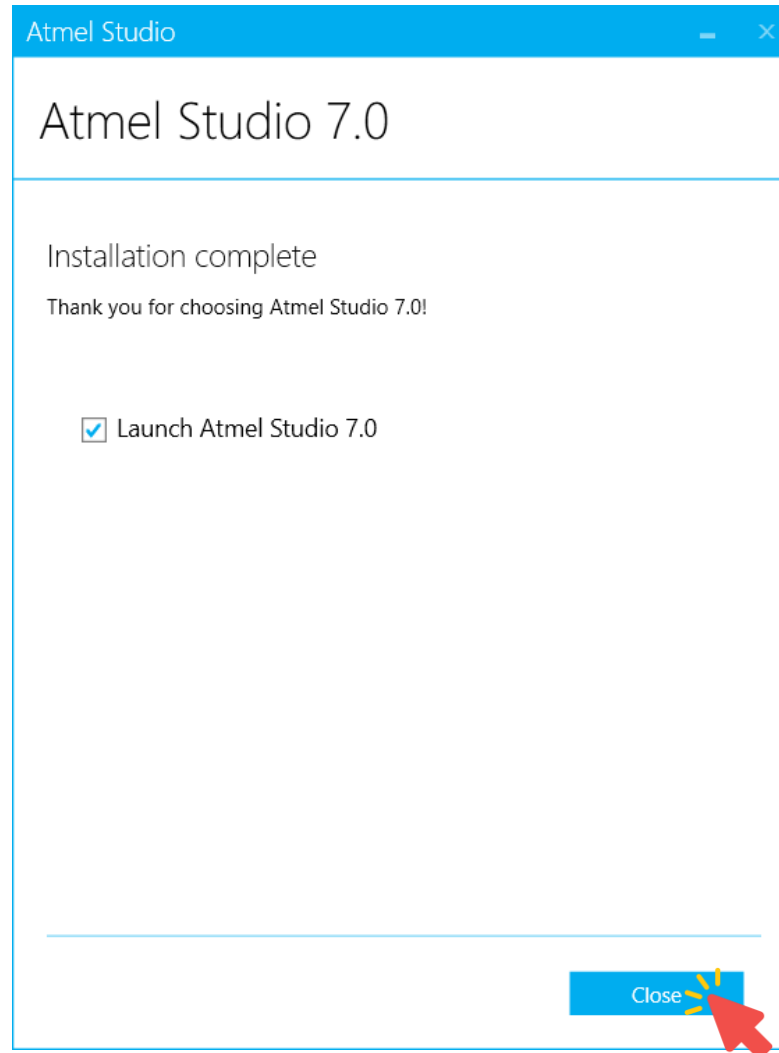
Atmel Studio 7: Setup Steps

11. Click Install.

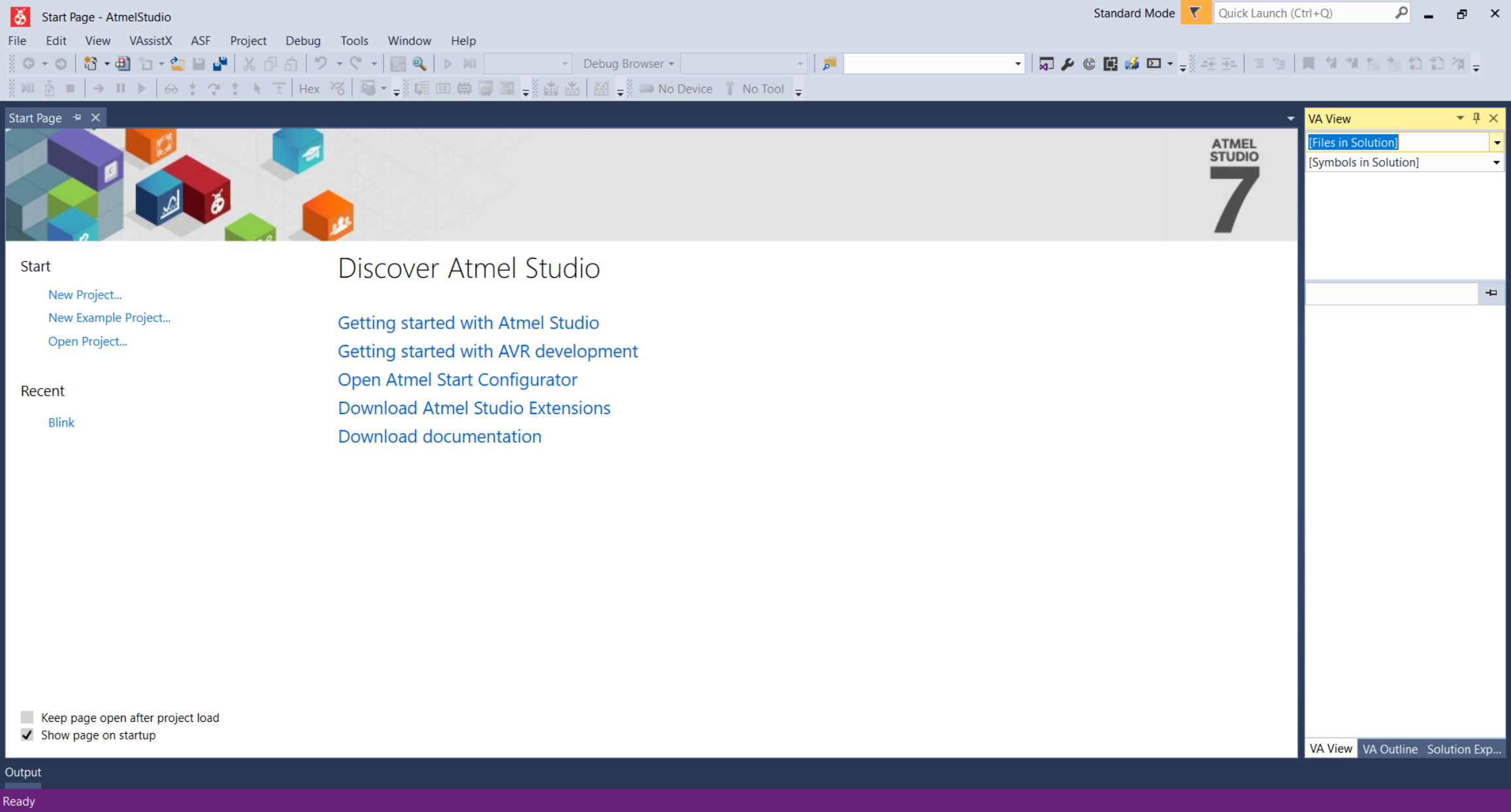


Atmel Studio 7: Setup Steps

12. Click Close.

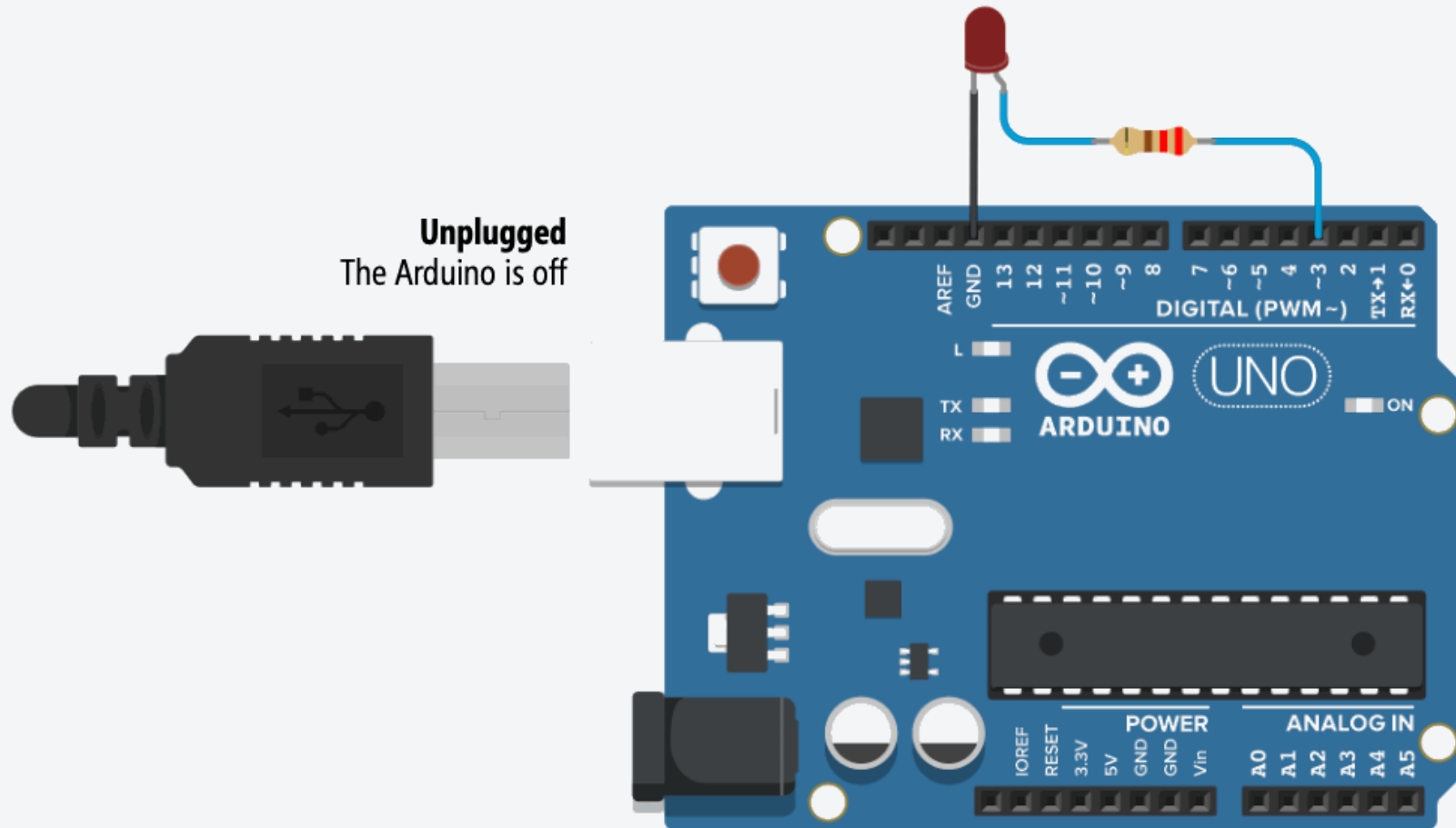


Atmel Studio 7: Start Page



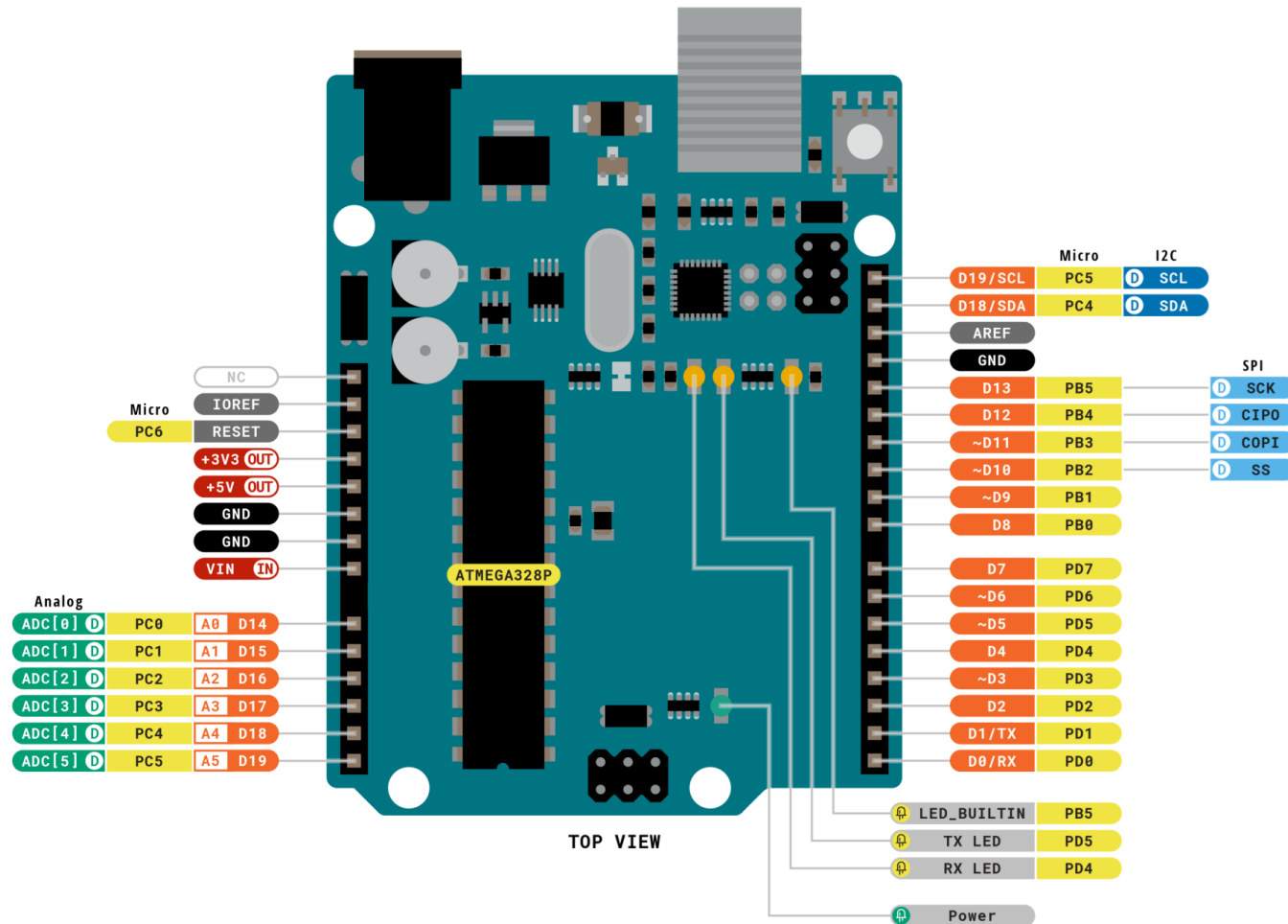
Your First AVR Project: Blinking an LED

- Turn an LED on and off every second.



Your First AVR Project: Idea

- The **I/O pin 13** in **Arduino Uno** connects directly to the **ATmega328P** microcontroller's **Port B, bit 5 pin**, labeled **PB5** in the diagram.

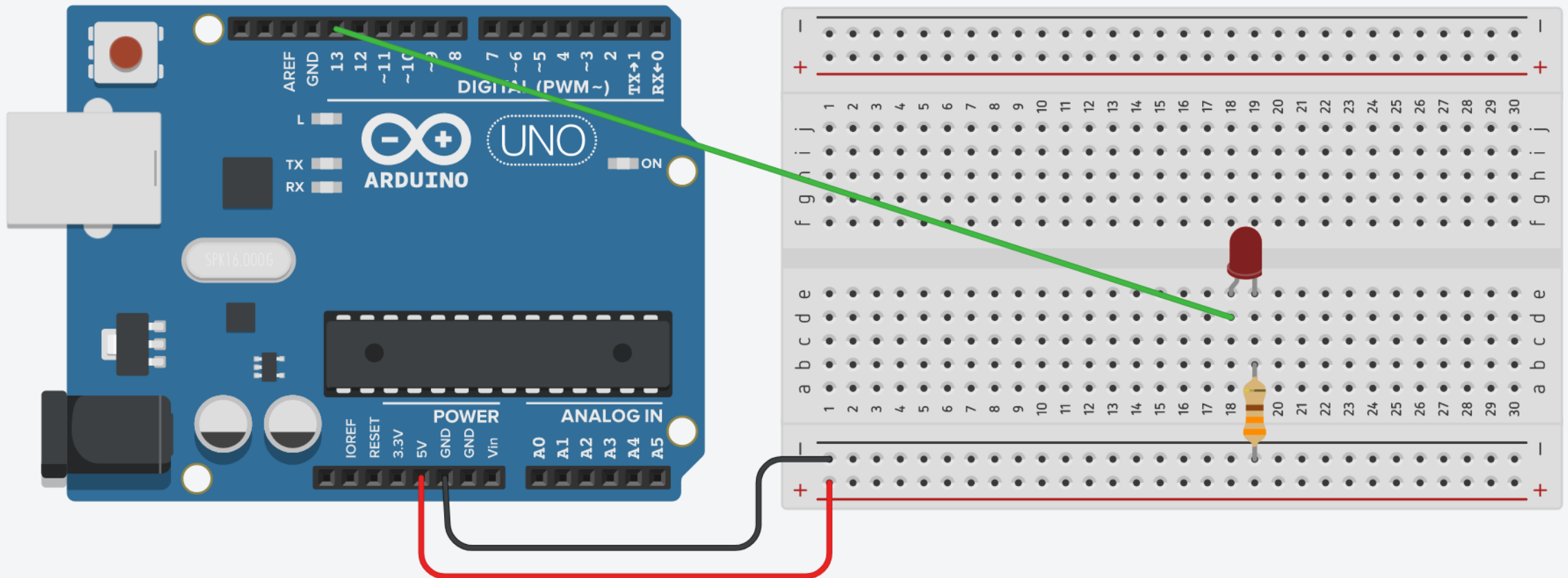


Your First AVR Project: Idea

- In this application, the LED is turned on for 1 second, and then the LED is turned off for 1 second.
- When we are programming in pure C, you need to define the exact pin where you want to write to.
- In our case, Arduino digital pin 13 is pin 5 of PORTB, or `0b00100000` in binary.

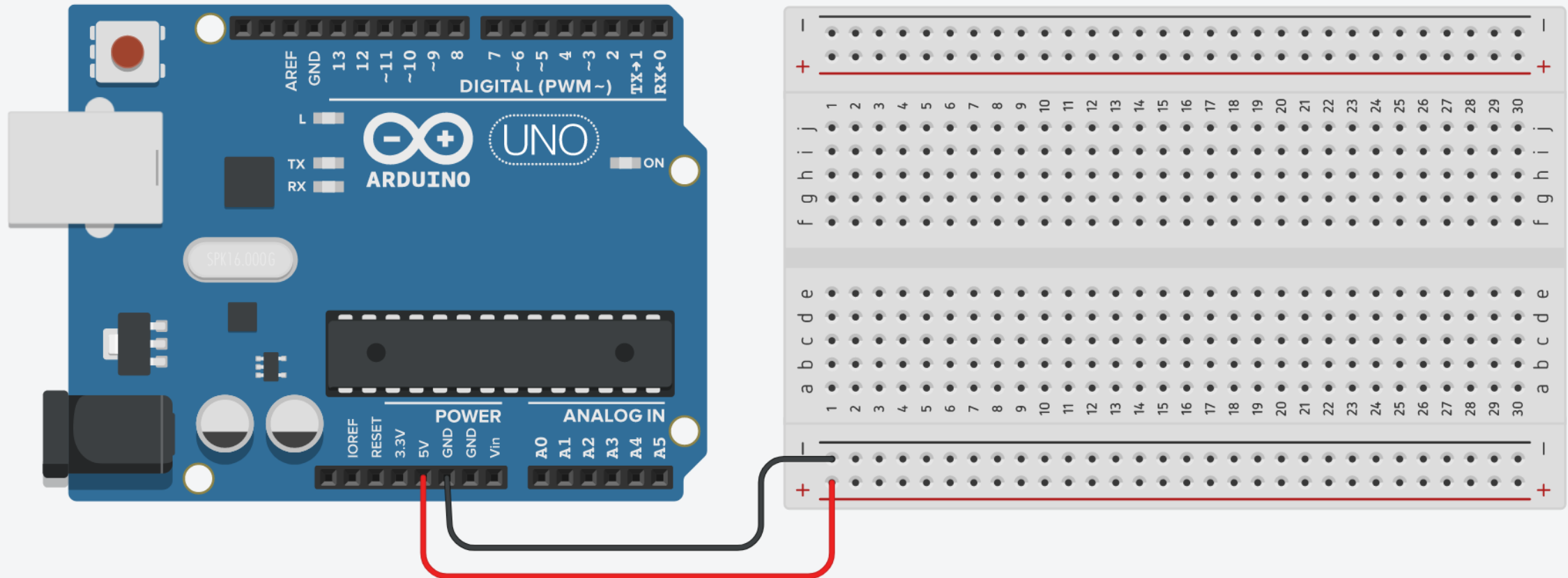
Your First AVR Project: Circuit

- Turn an LED on and off every second.



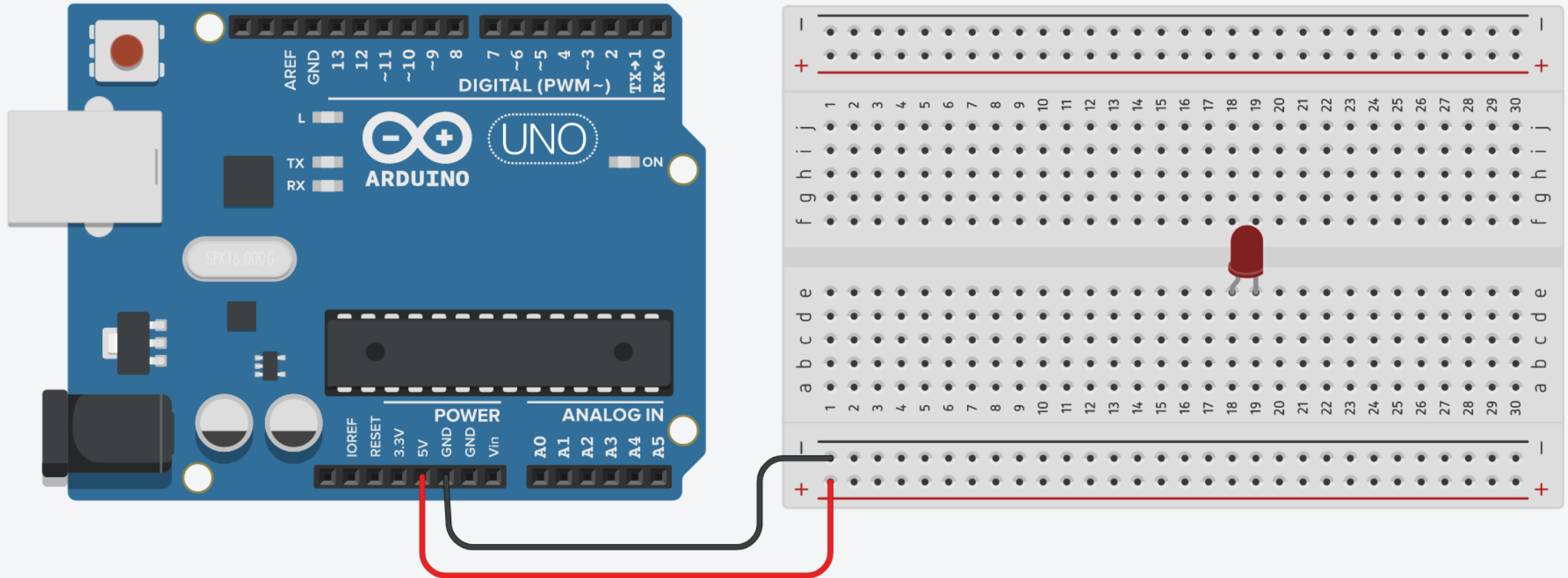
Your First AVR Project: Steps

1. Connect breadboard **power (+)** and **ground (-)** rails to Arduino **5V** and **ground (GND)**, respectively.



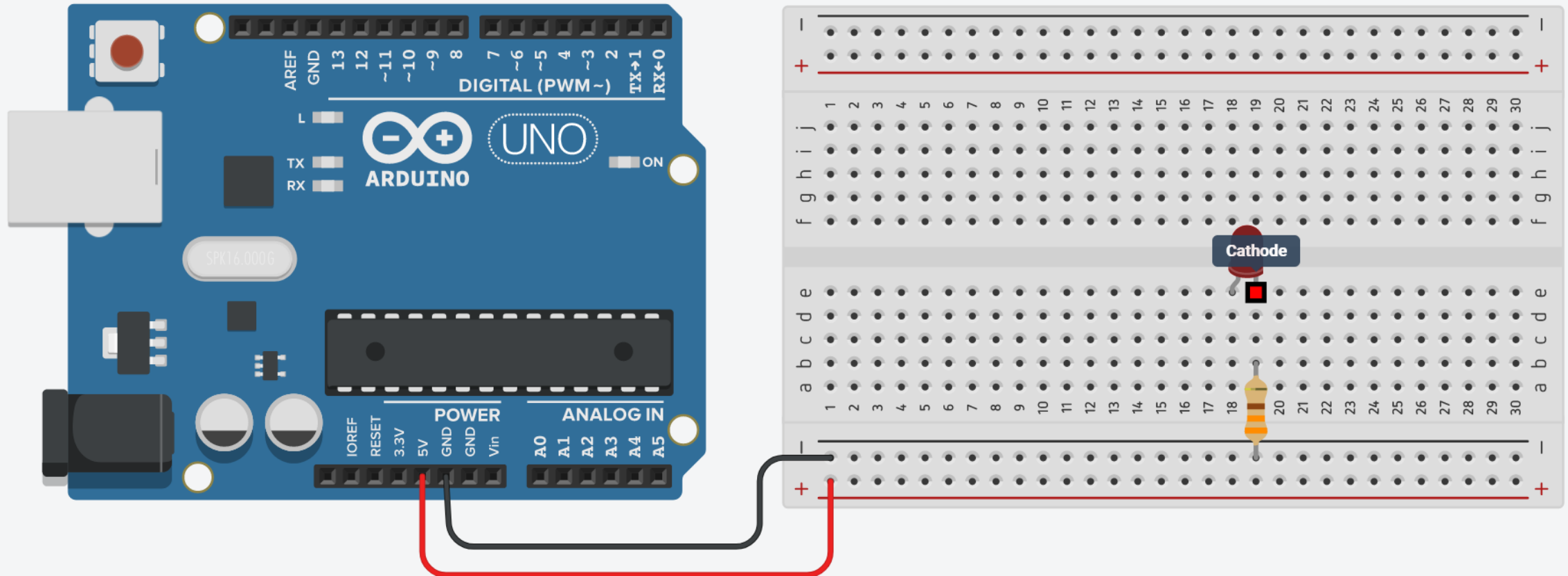
Your First AVR Project: Steps

2. Plug the **LED** into two different breadboard rows.



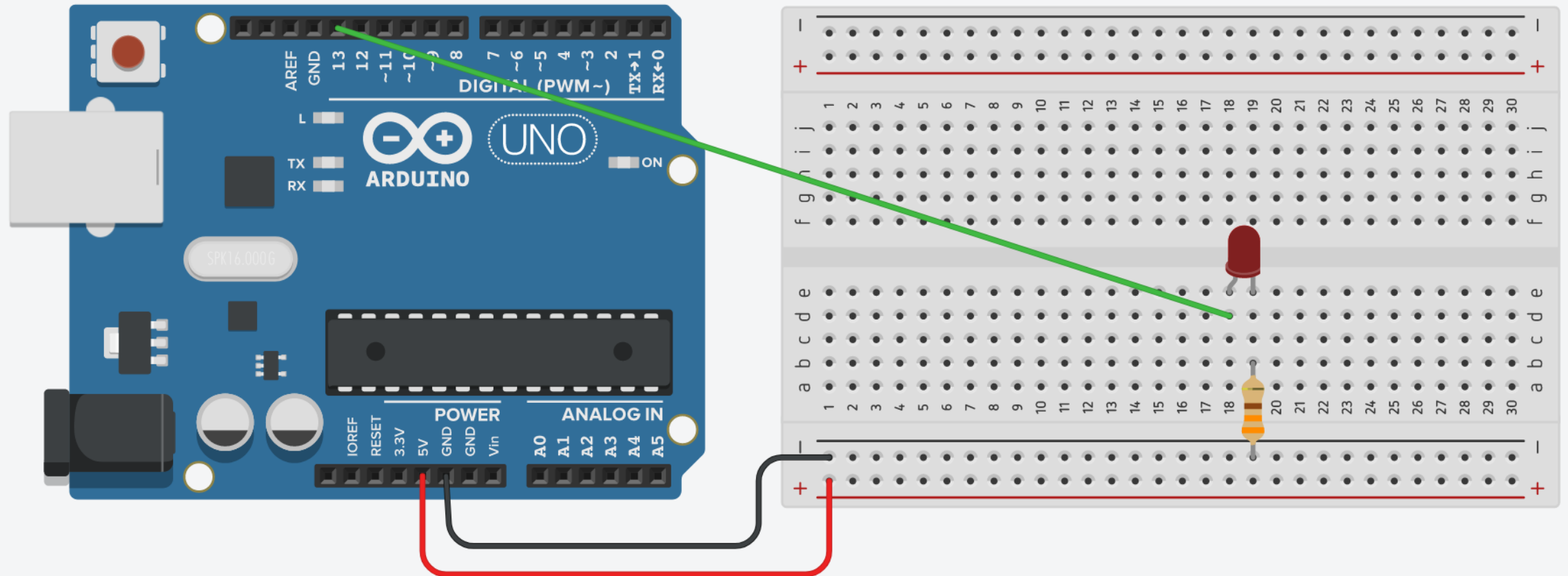
Your First AVR Project: Steps

- The **cathode (shorter leg)** connects to one leg of a **resistor of 330Ω** , and the **other resistor leg to the ground**.



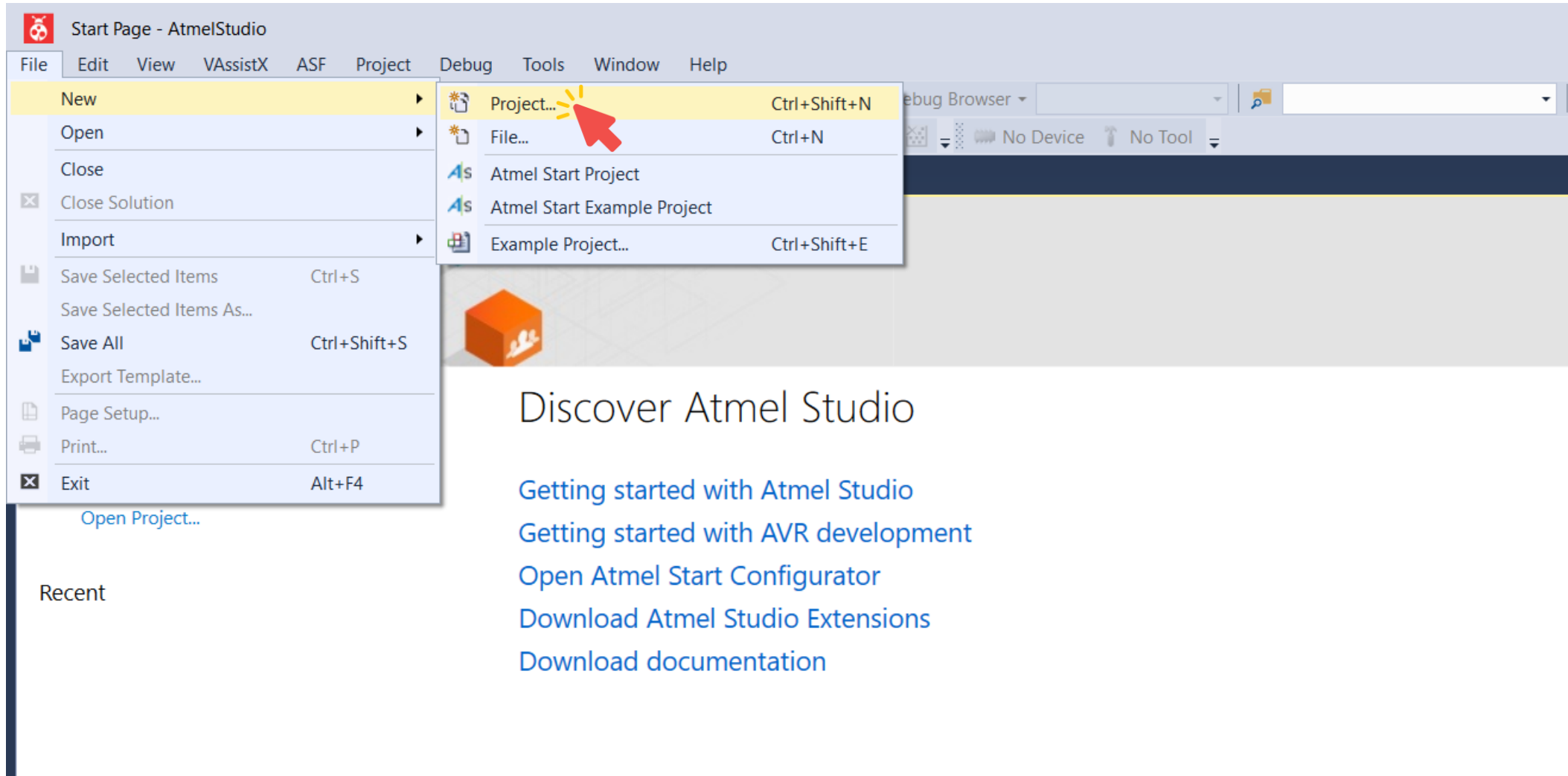
Your First AVR Project: Steps

4. Wire up the LED anode (longer leg) to Arduino **pin 13**.



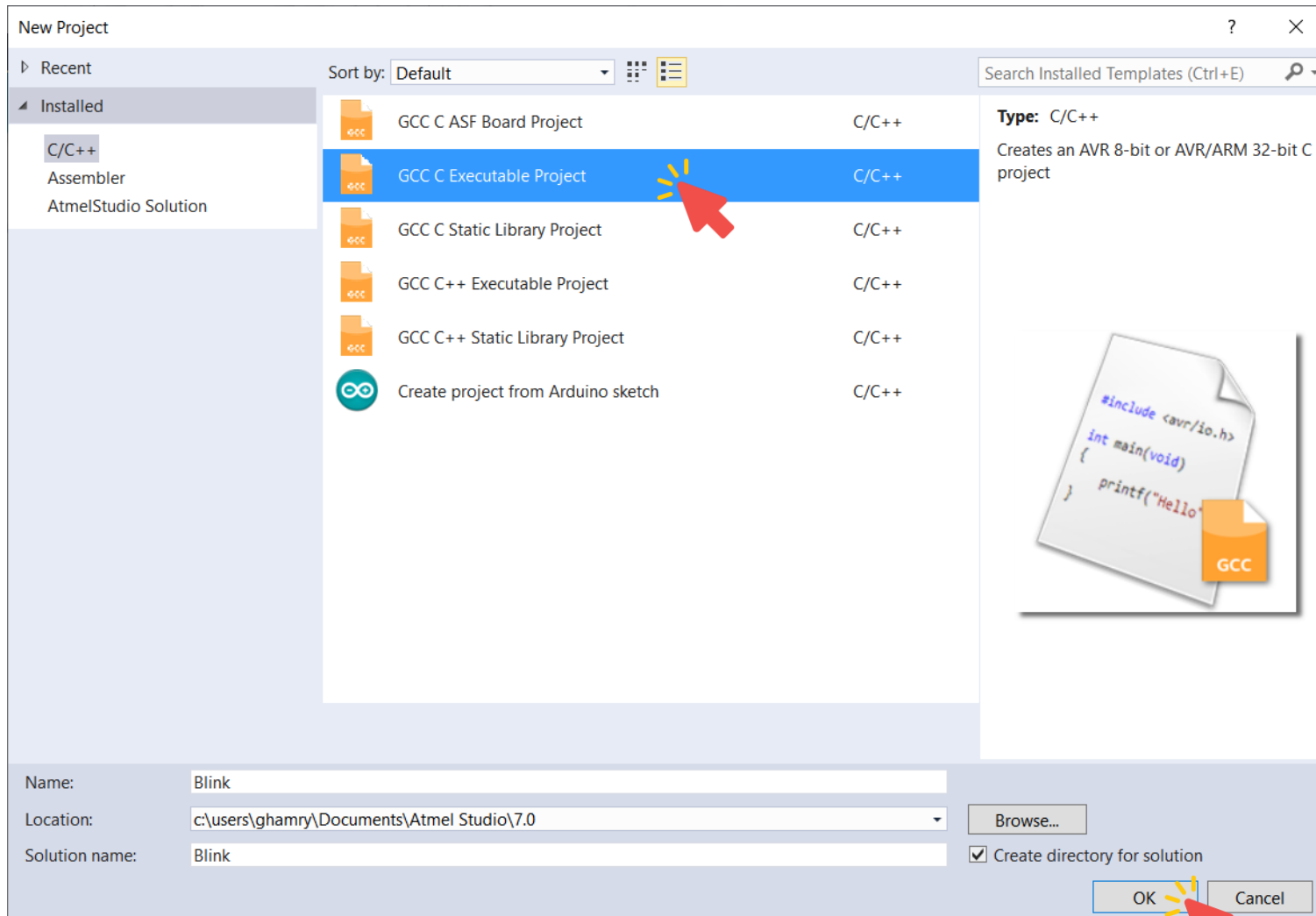
Your First AVR Project: New Project

1. Go to **File**, then **New**, and select **Project**.



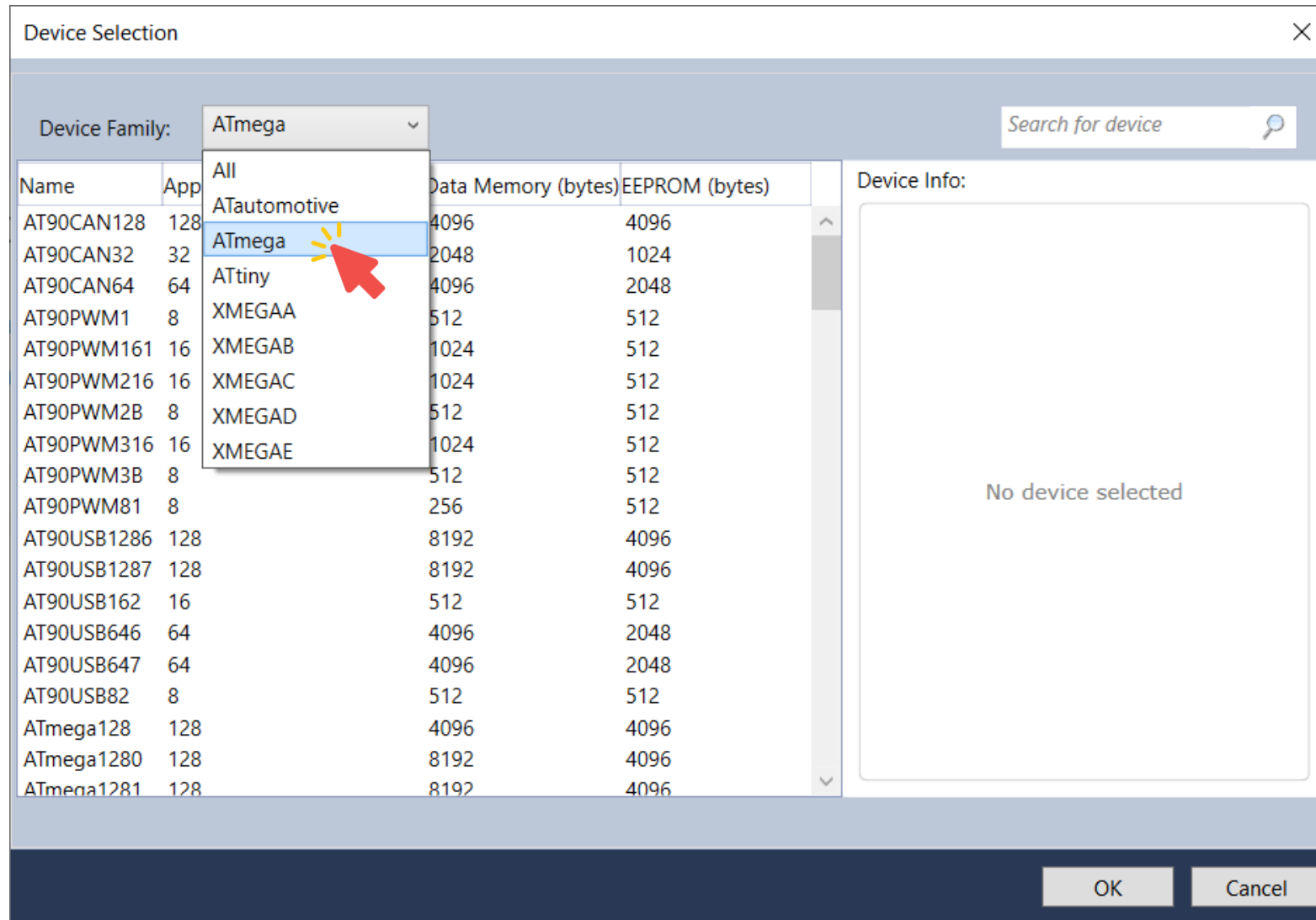
Your First AVR Project: New Project

2. Select **GCC C Executable File**, write the **Name** of your project, click **OK**.



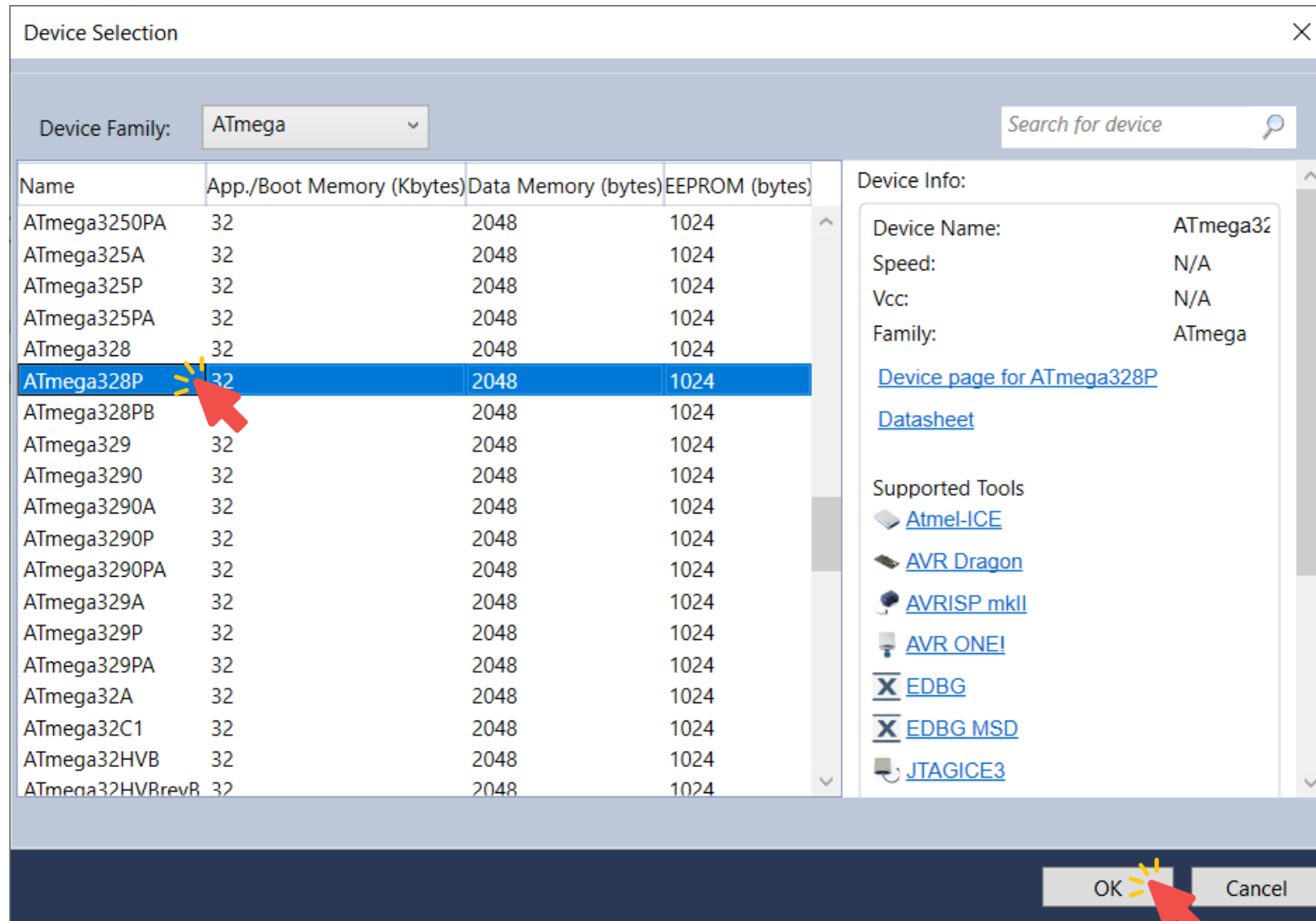
Your First AVR Project: New Project

3. From Drive Family, choose **ATmega**.



Your First AVR Project: New Project

4. Choose the microcontroller **ATmega328P**, and click **OK**.



Your First AVR Project: AVR Code

```
#define F_CPU 16000000 // 16 MHz Clock
#include <avr/io.h> // IO Ports Definition
#include <util/delay.h> // Delay function

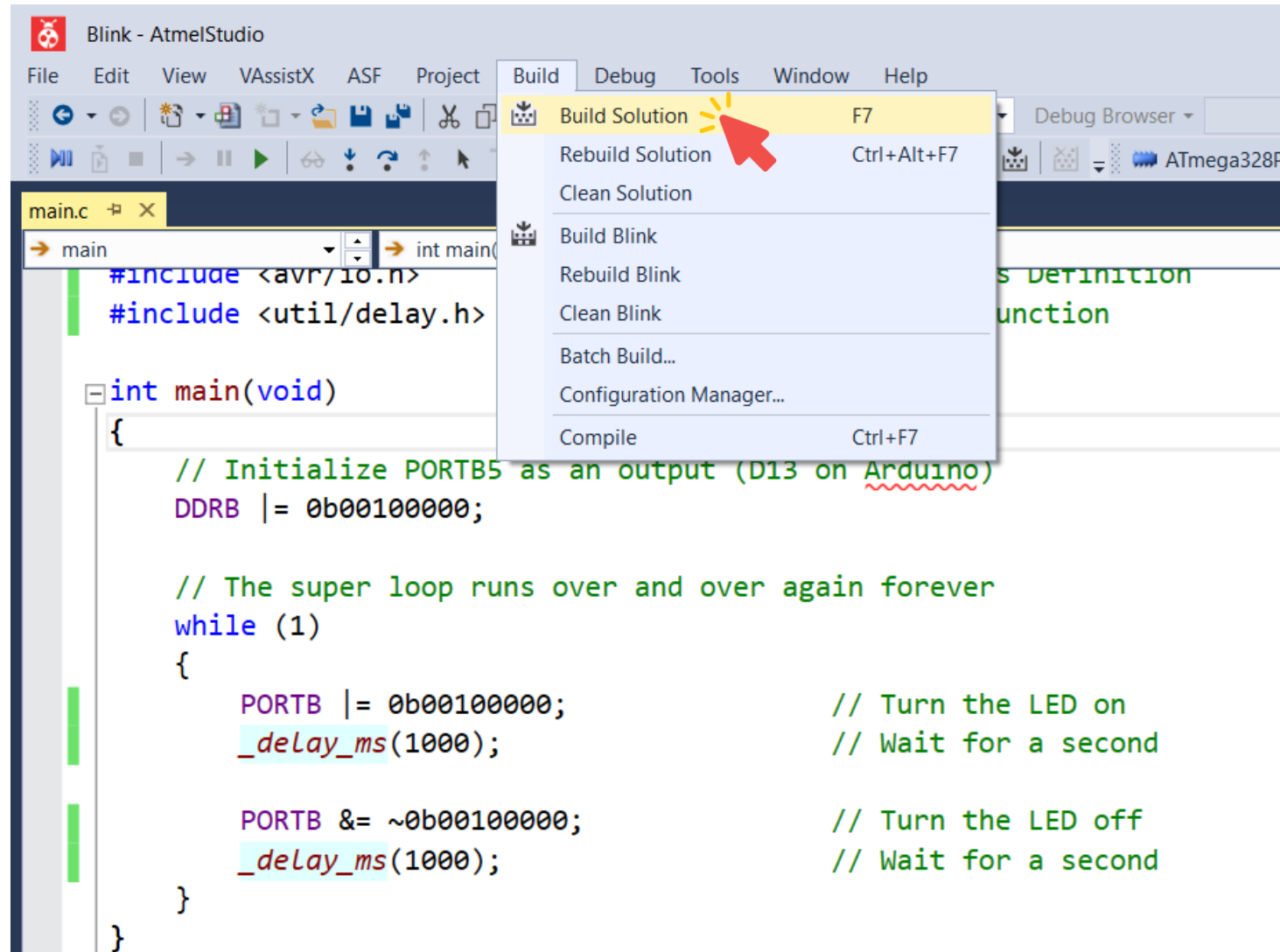
int main(void)
{
    // Initialize PORTB5 as an output (D13 on Arduino)
    DDRB |= 0b00100000;

    // The super loop runs over and over again forever
    while (1)
    {
        PORTB |= 0b00100000; // Turn the LED on
        _delay_ms(1000); // Wait for a second

        PORTB &= ~0b00100000; // Turn the LED off
        _delay_ms(1000); // Wait for a second
    }
}
```

Your First AVR Project: Building Solution

- Go to **Build**, and click **Build Solution**, or Press **F7** to **generate the HEX file**.



Your First AVR Project: New Project

- The contents of the **HEX file** after building the solution is

```
:10000000C9434000C943E000C943E000C943E0082
:10001000C943E000C943E000C943E000C943E0068
:10002000C943E000C943E000C943E000C943E0058
:10003000C943E000C943E000C943E000C943E0048
:10004000C943E000C943E000C943E000C943E0038
:10005000C943E000C943E000C943E000C943E0028
:10006000C943E000C943E0011241FBECFEFD8E04C
:10007000DEBFCDBF0E9440000C9456000C940000DF
:10008000259A2D9A2FEF83ED90E3215080409040E8
:10009000E1F700C000002D982FEF83ED90E3215091
:1000A00080409040E1F700C00000EBCFF894FFCF14
:00000001FF
```


Your First AVR Project: Discussion

	7	6	5	4	3	2	1	0
PORTB	0	0	0	0	0	0	0	0

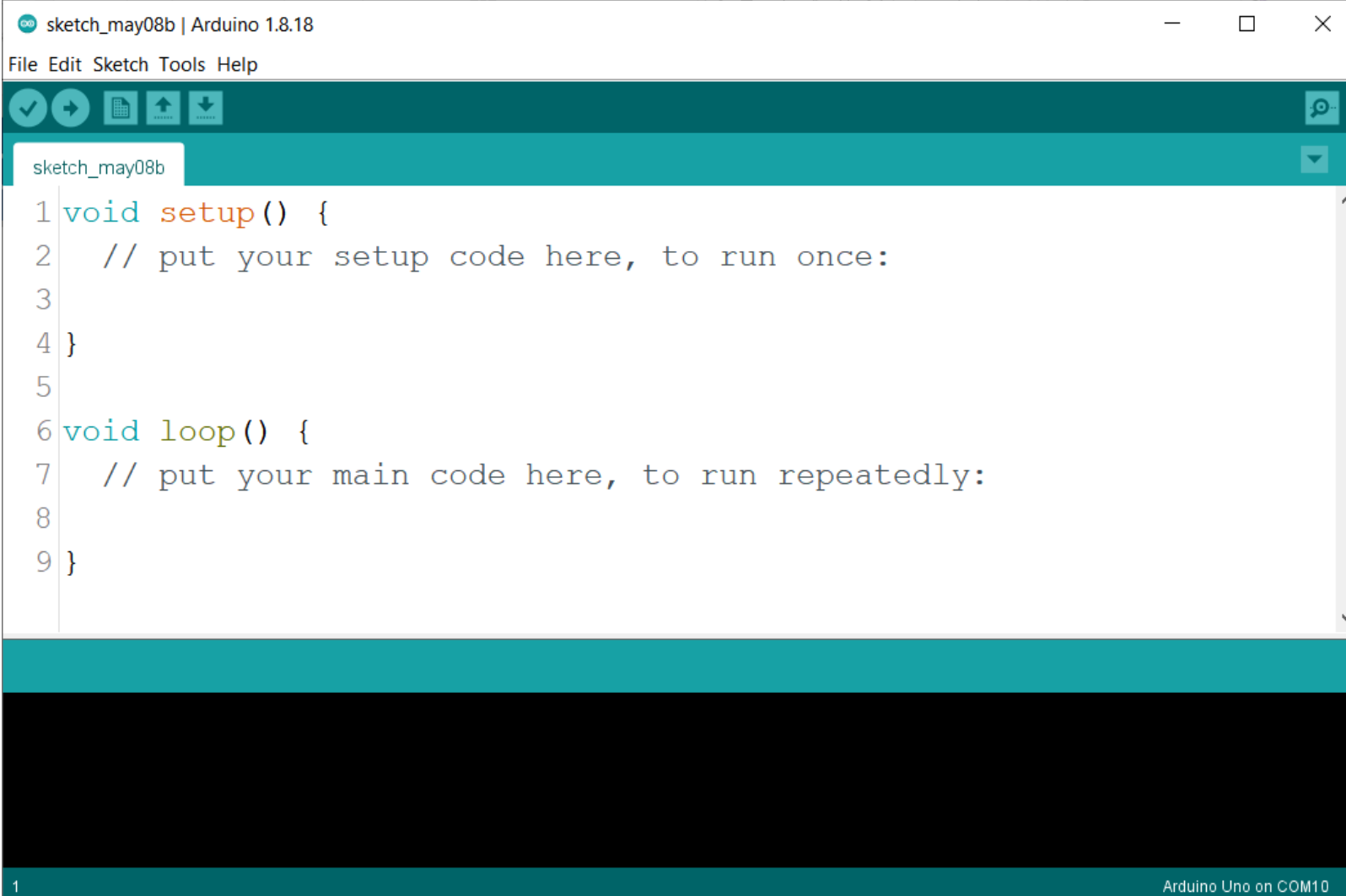
PORTB |= 0b00100000;

	7	6	5	4	3	2	1	0
OR	0	0	1	0	0	0	0	0

	7	6	5	4	3	2	1	0
PORTB	0	0	1	0	0	0	0	0

Configuration Steps for Programming Arduino Board

1. Open Arduino IDE.



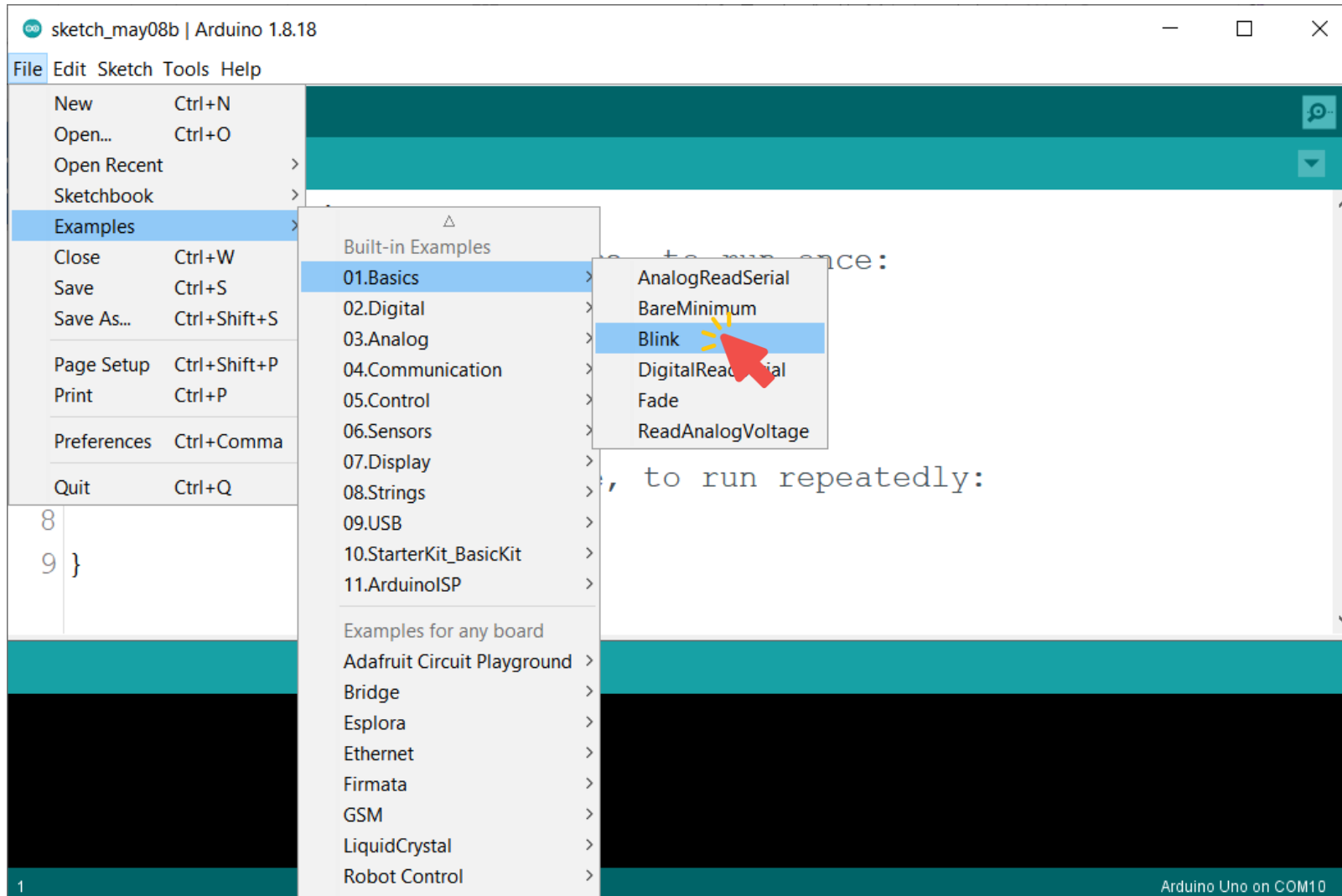
The screenshot displays the Arduino IDE window titled "sketch_may08b | Arduino 1.8.18". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for saving, running, uploading, and downloading. The sketch editor shows the following code:

```
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 }
```

The status bar at the bottom indicates "1" on the left and "Arduino Uno on COM10" on the right.

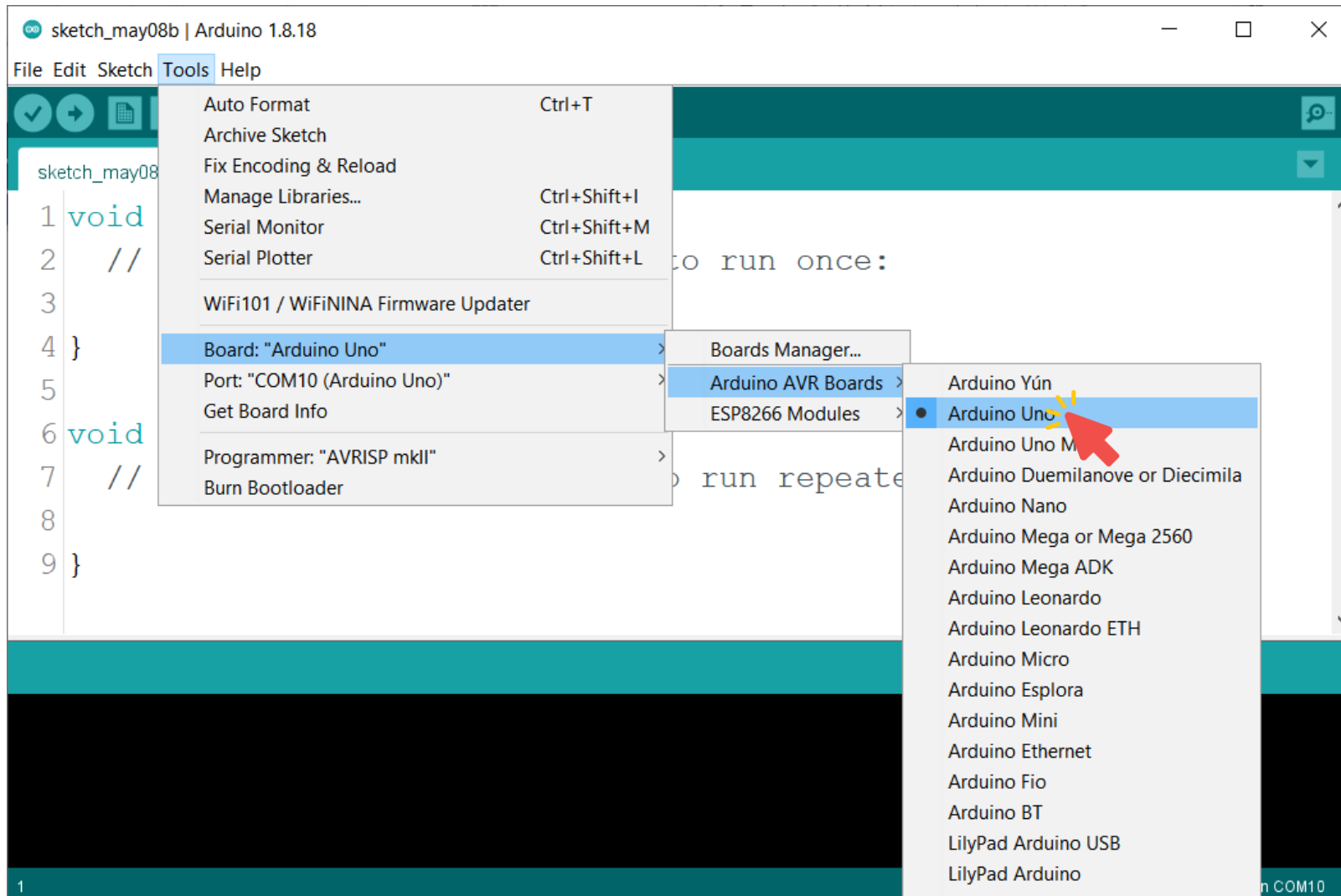
Configuration Steps for Programming Arduino Board

2. Go to **File**, then **Examples**, then **01. Basics**, and choose **Blink**.



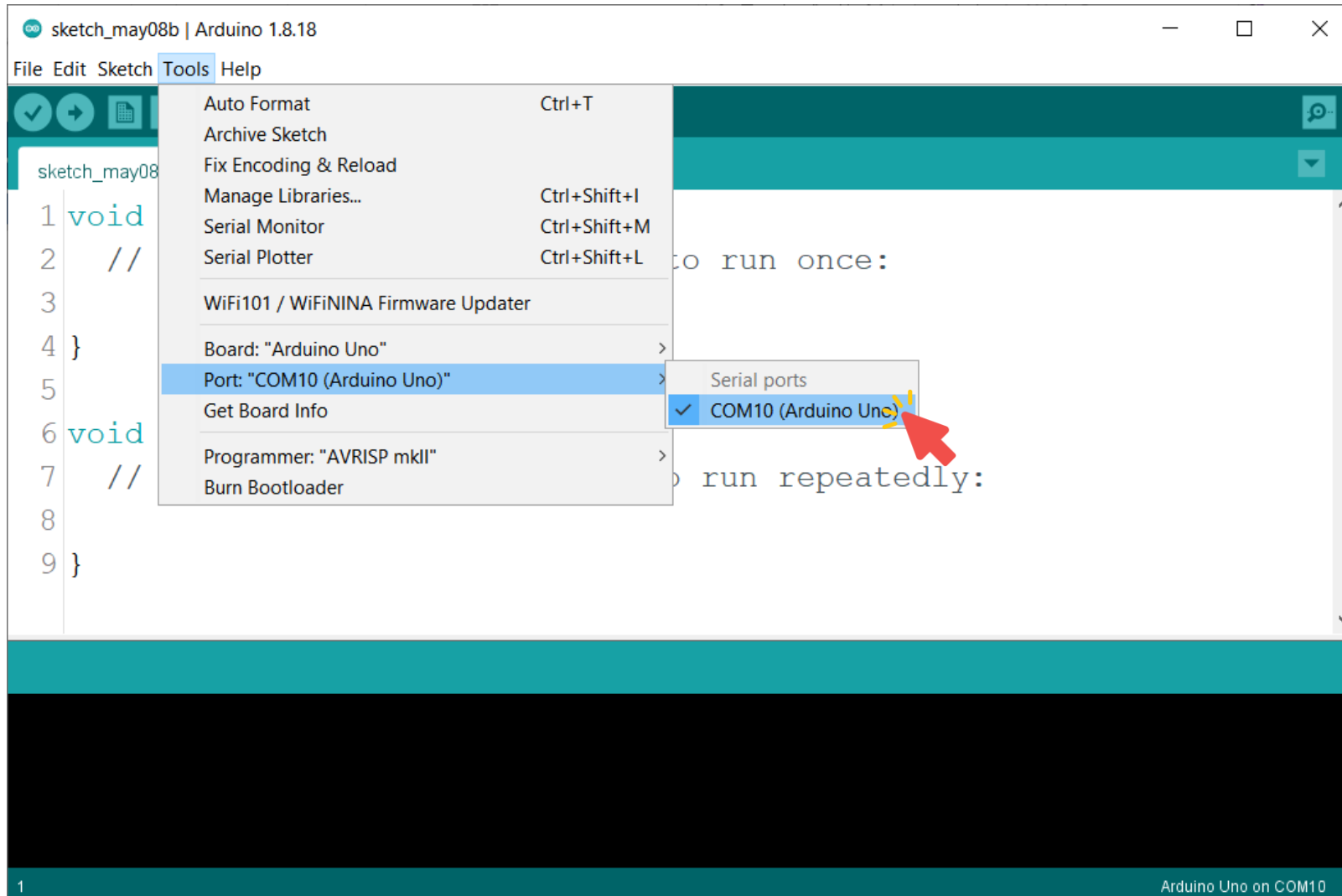
Configuration Steps for Programming Arduino Board

3. Go to **Tools**, then **Board**, and choose **Arduino Uno**.



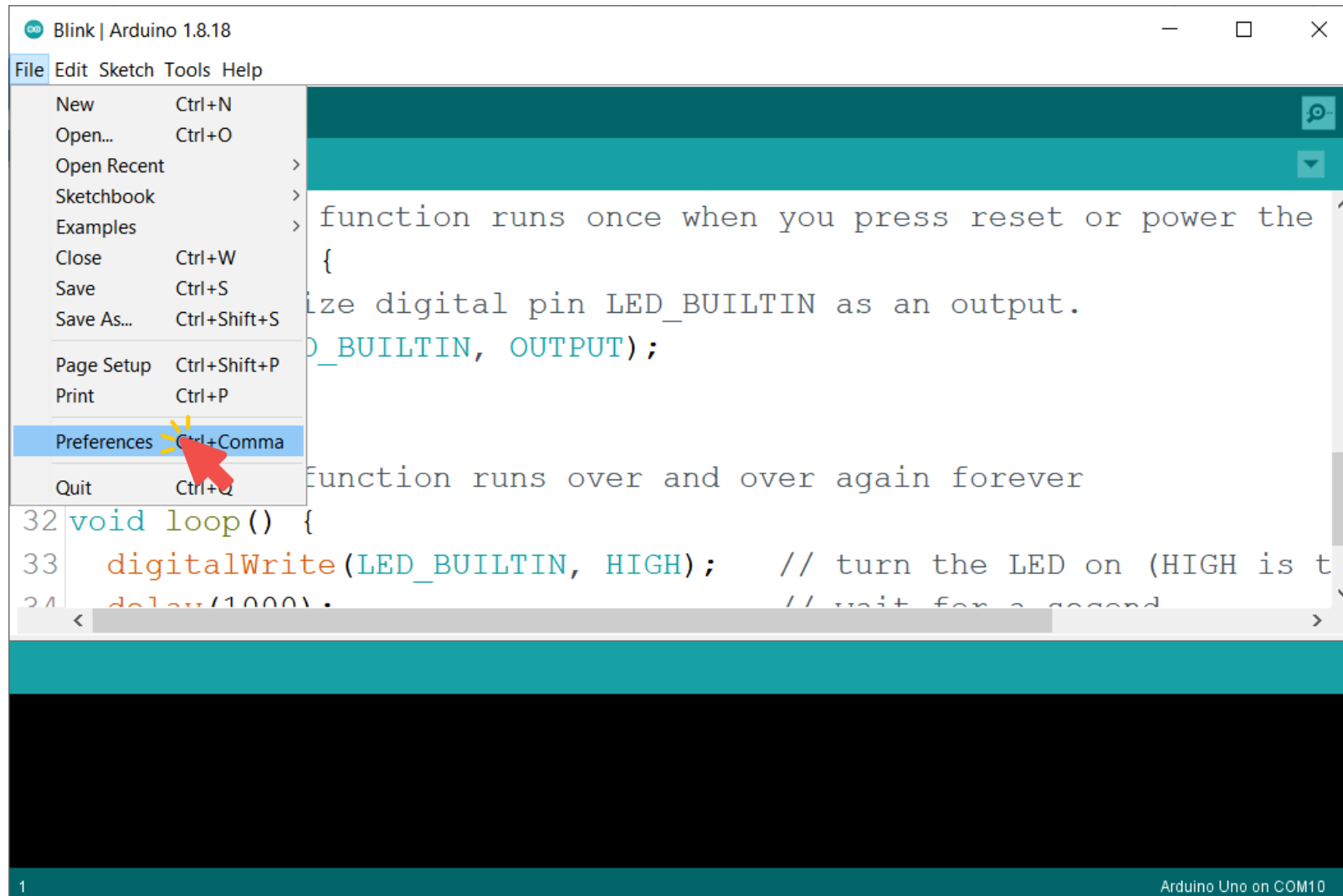
Configuration Steps for Programming Arduino Board

4. Go to **Tools**, then **Port**, the **COM**.



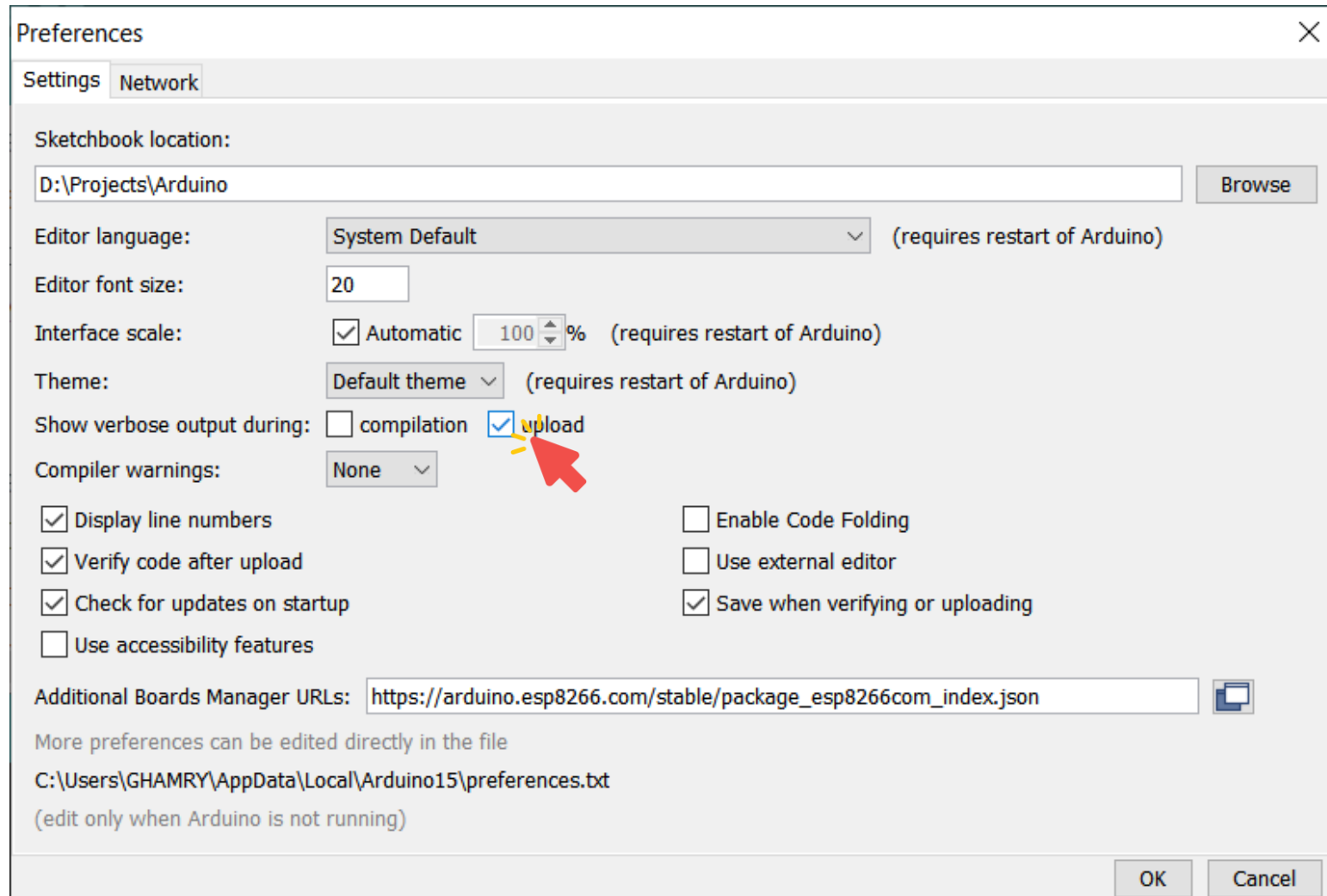
Configuration Steps for Programming Arduino Board

5. Go to **File**, and choose **Preferences**.



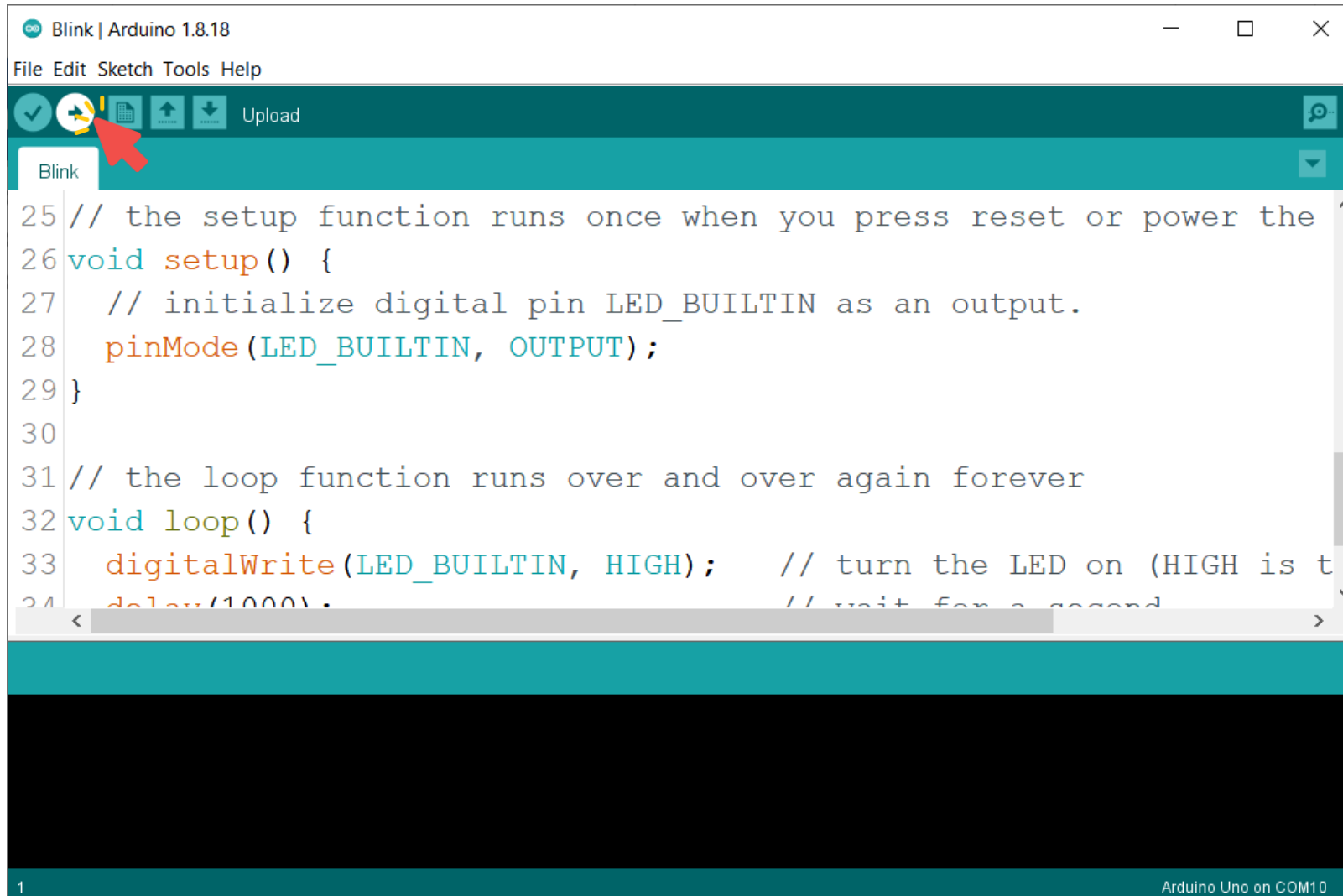
Configuration Steps for Programming Arduino Board

6. Check on Show verbose output during: Upload.



Configuration Steps for Programming Arduino Board

7. Upload the code to Arduino Uno.



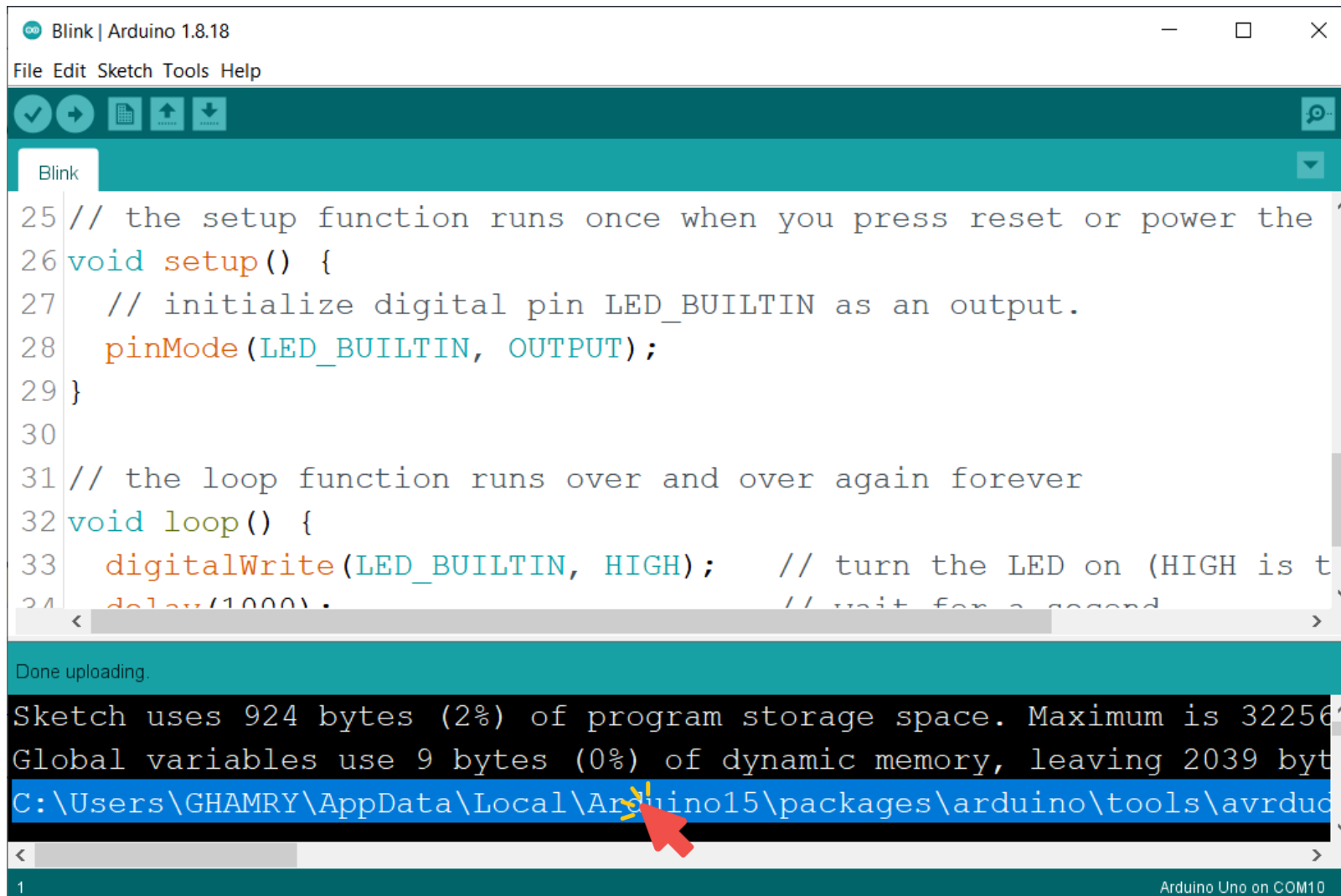
The screenshot shows the Arduino IDE interface. The title bar reads "Blink | Arduino 1.8.18". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains several icons, with the "Upload" icon (a yellow arrow pointing right) highlighted by a red arrow. Below the toolbar, the sketch name "Blink" is visible. The main text area contains the following code:

```
25 // the setup function runs once when you press reset or power the
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is t
34   delay(1000); // wait for a second
```

The status bar at the bottom right indicates "Arduino Uno on COM10".

Configuration Steps for Programming Arduino Board

8. Copy the upload command.



The screenshot shows the Arduino IDE interface. The main window displays the code for a Blink sketch. Below the code editor, the status bar shows the upload command path: `C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdu`. A red arrow points to this path, indicating it should be copied.

```
25 // the setup function runs once when you press reset or power the
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is t
34   delay(1000); // wait for a second
```

Done uploading.

Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 byt
C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrduc

1 Arduino Uno on COM10

Configuration Steps for Programming Arduino Board

- The copied `upload command` is

```
C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude -
CC:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/etc/avrdude.conf -v
-patmega328p -carduino -PCOM10 -b115200 -D -
Uflash:w:C:\Users\GHAMRY\AppData\Local\Temp\arduino_build_889169/Blink.ino.hex:i
```

Configuration Steps for Programming Arduino Board

- From the `upload command`, the `avrdude` location is

```
C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude
```

- Add `.exe` to it.

```
C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude.exe
```

Configuration Steps for Programming Arduino Board

- From the **upload command**, the **arguments** are

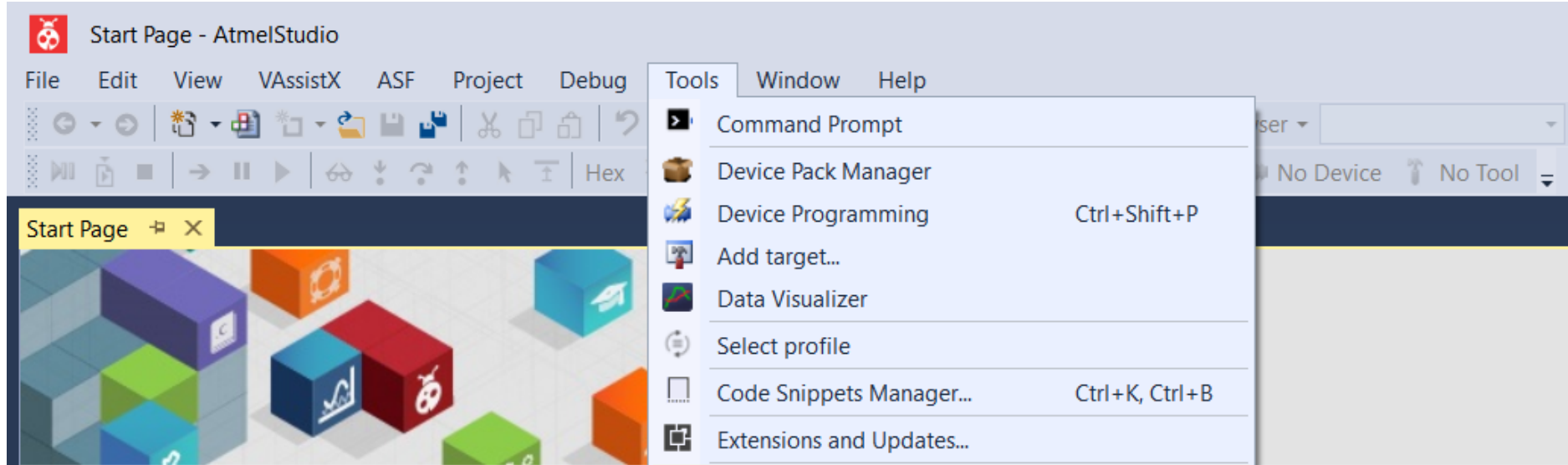
```
-CC:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-  
arduino17/etc/avrdude.conf -v -patmega328p -carduino -PCOM10 -b115200 -D -  
Uflash:w:C:\Users\GHAMRY\AppData\Local\Temp\arduino_build_889169/Blink.ino.hex:i
```

- Modify the arguments to

```
-C "C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-  
arduino17/etc/avrdude.conf" -v -patmega328p -carduino -PCOM10 -b115200 -D -  
Uflash:w: "$(ProjectDir)Debug/$(TargetName).hex" :i
```

Configuration Steps for Programming Arduino Board

9. Go to **Tools**, and click **External Tools**.



The screenshot shows the Atmel Studio interface. The 'Tools' menu is open, and 'External Tools...' is highlighted with a red arrow. The menu items are:

- Command Prompt
- Device Pack Manager
- Device Programming (Ctrl+Shift+P)
- Add target...
- Data Visualizer
- Select profile
- Code Snippets Manager... (Ctrl+K, Ctrl+B)
- Extensions and Updates...
- External Tools...**
- Import and Export Settings...
- Customize...
- Options...

The background shows the 'Start Page' with options like 'New Project...', 'New Example Project...', and 'Open Project...'.

[Getting started with AVR development](#)

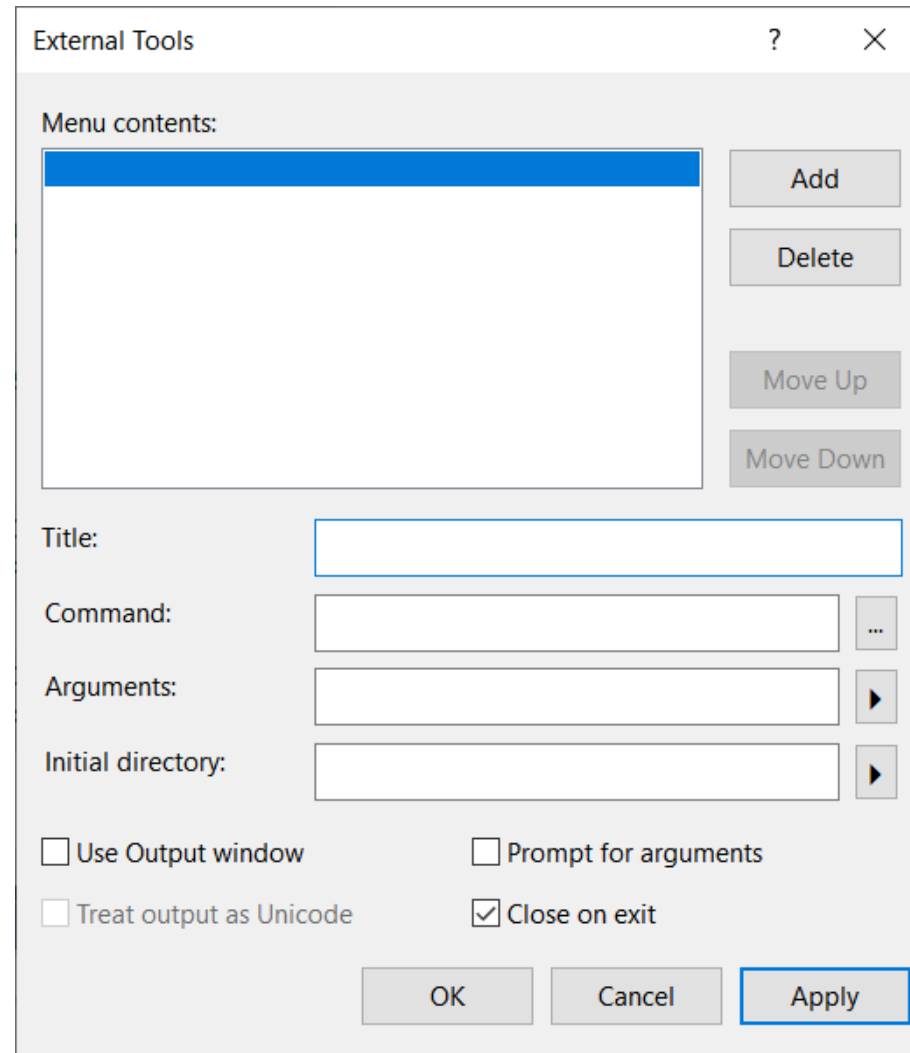
[Open Atmel Start Configurator](#)

[Download Atmel Studio Extensions](#)

[Download documentation](#)

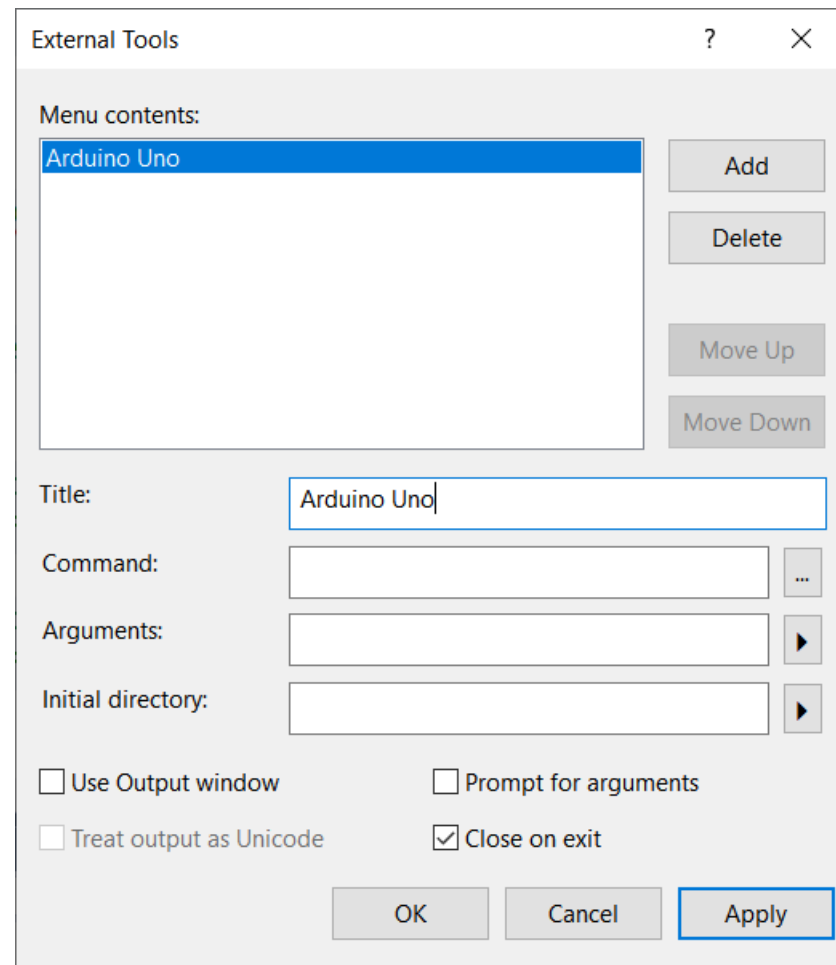
Configuration Steps for Programming Arduino Board

10. This window will show up.



Configuration Steps for Programming Arduino Board

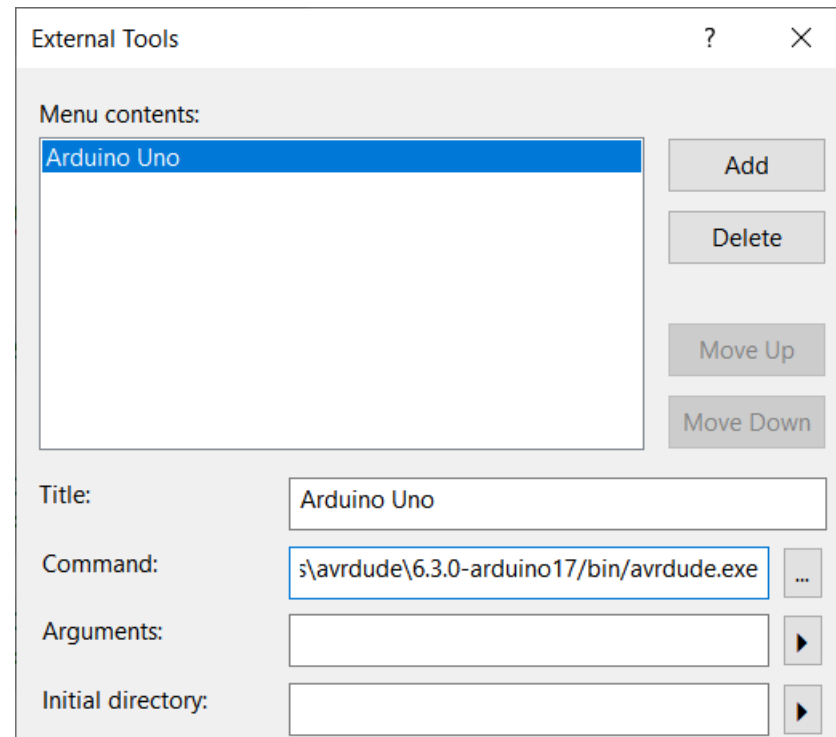
11. Click **Add** to add a new Tool.
 - For **Title**, write **Arduino Uno** for example.



Configuration Steps for Programming Arduino Board

- For **Command**, write the path to **avrdude.exe** that will be in the location where you have **installed the Arduino IDE**.

`C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/bin/avrdude.exe`



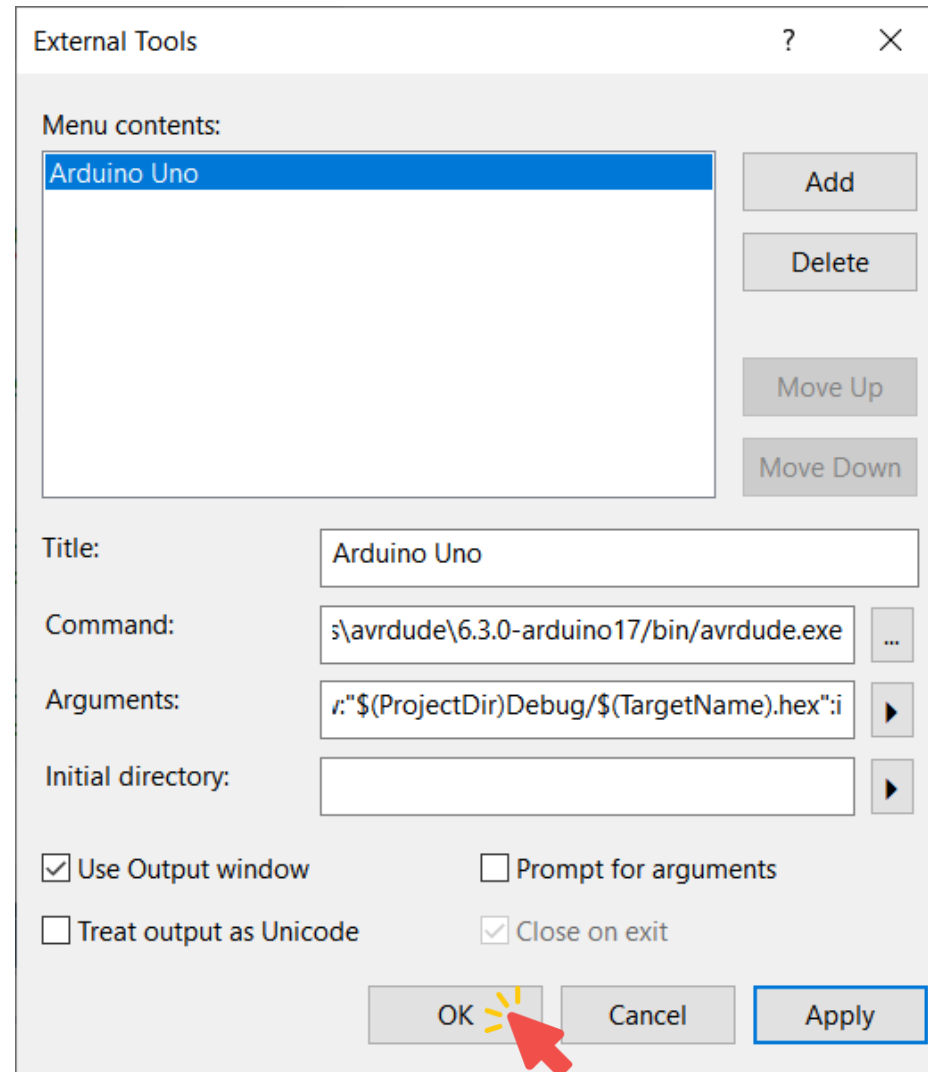
Configuration Steps for Programming Arduino Board

- For Arguments, it should have the 3 most important parameters, the **Microcontroller** which is dependent on the Arduino board you are using, **COM Port**, and **Baud Rate**.
- COM Port system is dependent and can be determined from Device Manager.
- Baud Rate should be 115200.

```
-C"C:\Users\GHAMRY\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-  
arduino17/etc/avrdude.conf" -v -patmega328p -carduino -PCOM10 -b115200 -D -  
Uflash:w:"$(ProjectDir)Debug/$(TargetName).hex":i
```

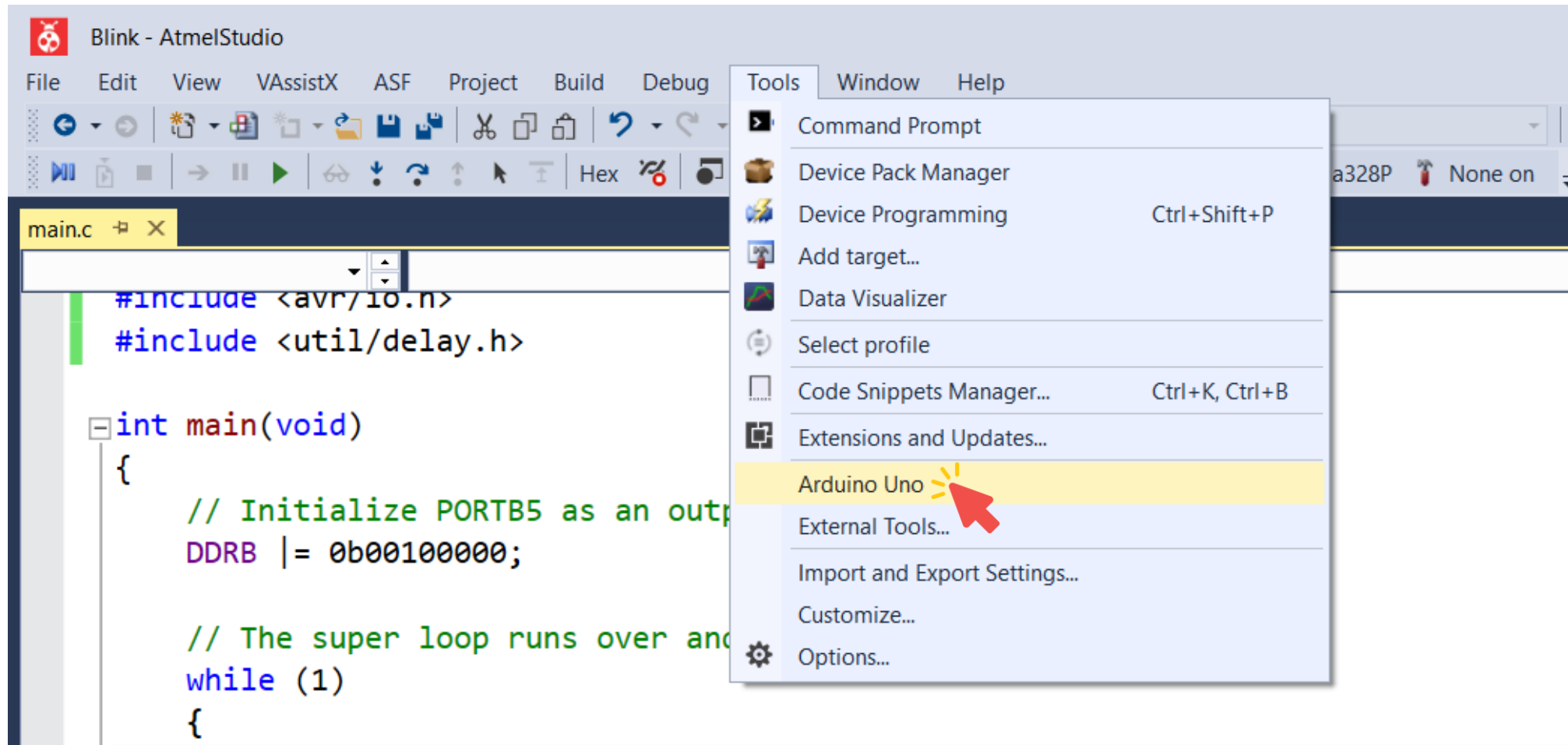
Configuration Steps for Programming Arduino Board

12. Check **Use output window**, and click **OK**.



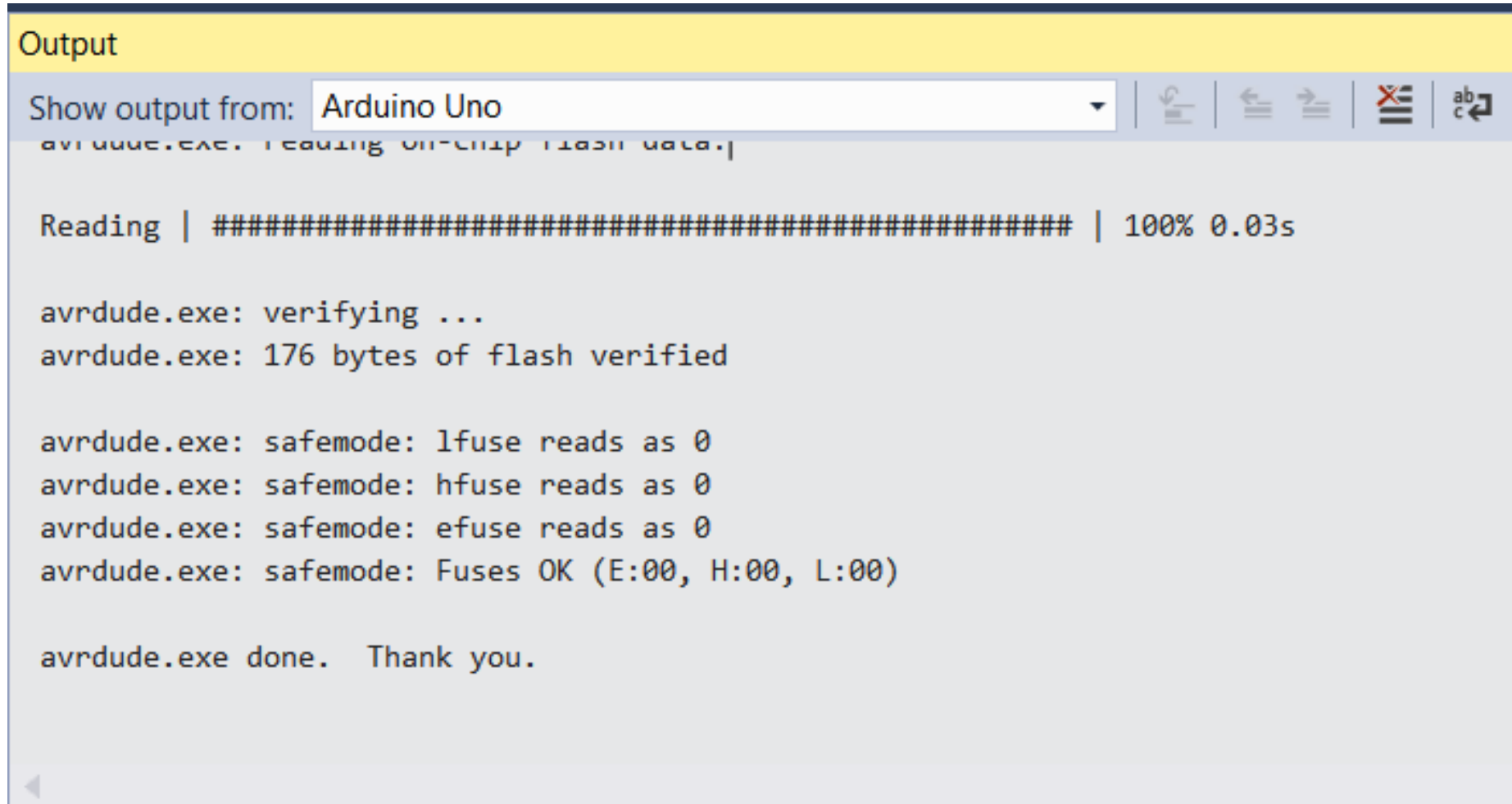
Configuration Steps for Programming Arduino Board

13. Go to **Tools**, and click **Arduino Uno**.



Configuration Steps for Programming Arduino Board

- If everything is fine, you should get a message like this



```
Output
Show output from: Arduino Uno
avrdude.exe: reading on-chip flash data.

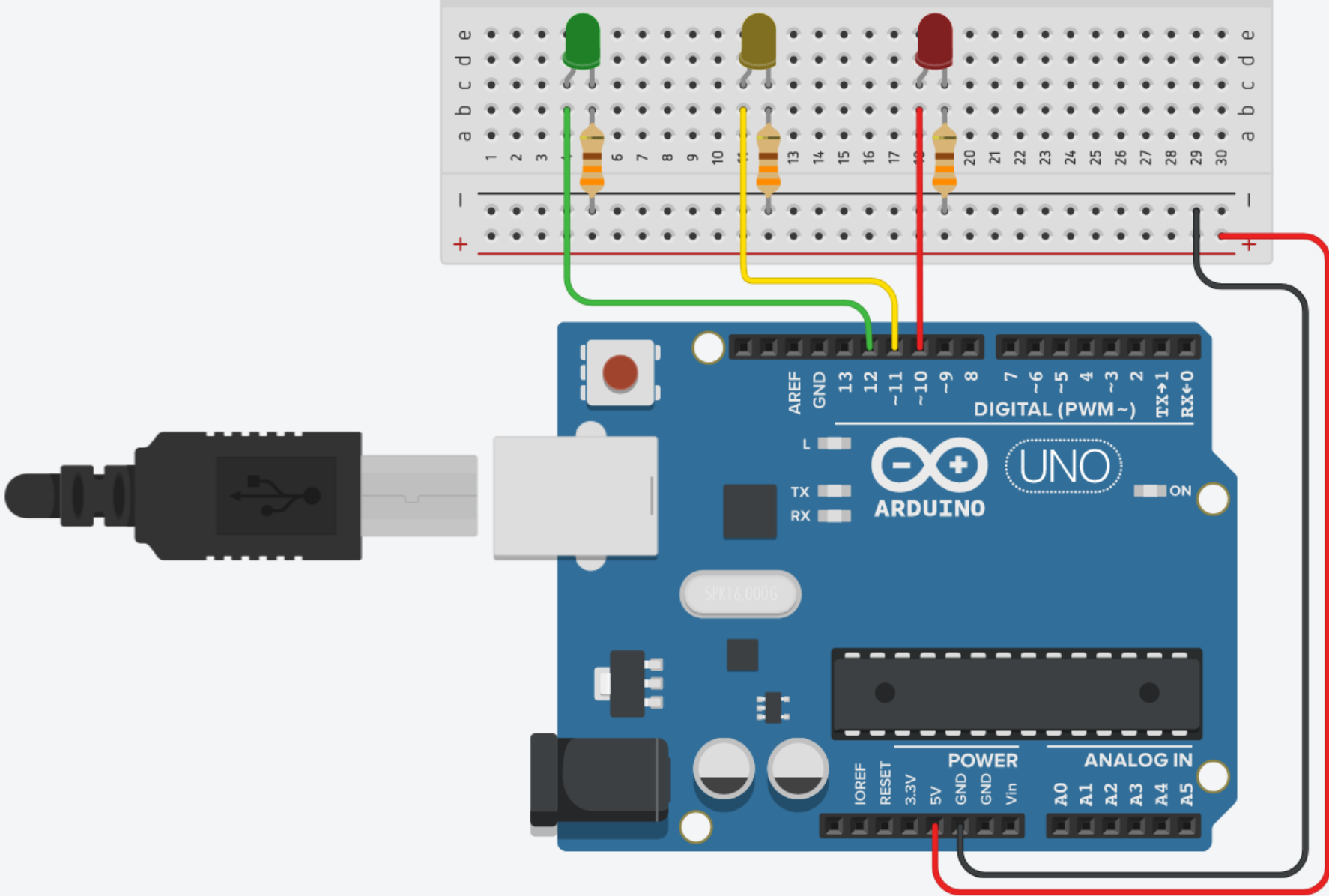
Reading | ##### | 100% 0.03s

avrdude.exe: verifying ...
avrdude.exe: 176 bytes of flash verified

avrdude.exe: safemode: lfuse reads as 0
avrdude.exe: safemode: hfuse reads as 0
avrdude.exe: safemode: efuse reads as 0
avrdude.exe: safemode: Fuses OK (E:00, H:00, L:00)

avrdude.exe done. Thank you.
```

Assignment 07: Traffic Lights



References

- [Atmel Studio 7 - Programming the Arduino Uno via the Bootloader Without Programmer.](#)
- [Arduino Uno Rev3](#)
- [Arduino Mega 2560 Rev3](#)
- [What You Need to Know About the ATmega328P](#)
- [ATmega168/328P-Arduino Pin Mapping](#)
- [ATmega2560-Arduino Pin Mapping](#)
- [Von-Neumann vs Harvard Architecture](#)

Useful Channels

- [Arafa Microsys - YouTube](#)
- [Khaled Magdy - YouTube](#)
- [Hashim EduTech - YouTube](#)

Useful Courses

- [Embedded C Atmel Studio - Arafa Microsys](#)
- [Embedded Systems Course - Khaled Magdy](#)
- [Learn Microcontroller and Atmel Studio AVR](#)