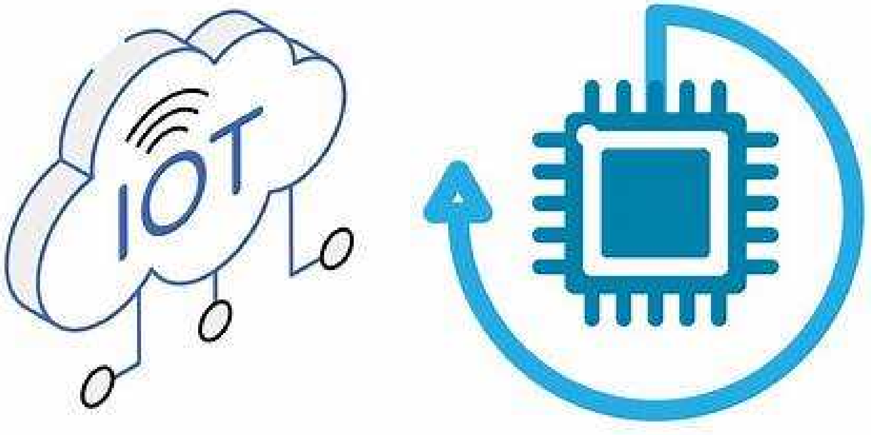
한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Internet of Things (IoT) Systems



Lecture 04

Microcontroller and Arduino

Ikram Syed, Ph.D.
Associate Professor
Department of Information and Communication Engineering
Hankuk University of Foreign Studies (HUFS)

Spring – 2025

Introduction to Embedded Systems

- Computers we are familiar with:
  - Desktops
  - Laptops
  - Servers
  - Mobile phones

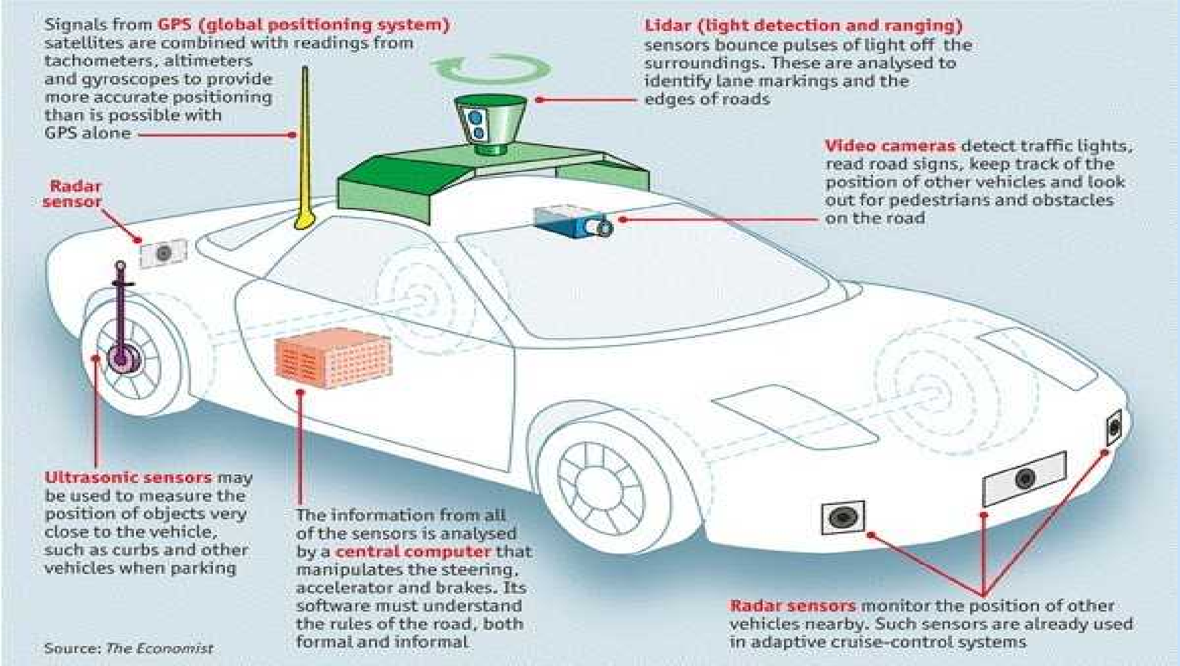
Introduction to Embedded Systems

- Computers that are often hidden in environment for which they are created
  - Refrigerator
  - Air conditioner
  - MR System

Introduction to Embedded Systems

- An embedded system is a computer system that has a dedicated function within a larger mechanical or electrical or biological system
- Embedded within other system
- Computers other than desktop, laptop, and server machines
- A microcontroller-based system that is designed to control a function or range of functions, and is not meant to be programmed by the end user
  - User may make choices concerning the functionality but cannot change them
  - User cannot make modifications to the software or program

Introduction to Embedded Systems



Signals from **GPS (global positioning system)** satellites are combined with readings from tachometers, altimeters and gyroscopes to provide more accurate positioning than is possible with GPS alone

Lidar (light detection and ranging) sensors bounce pulses of light off the surroundings. These are analysed to identify lane markings and the edges of roads

Radar sensor

Video cameras detect traffic lights, read road signs, keep track of the position of other vehicles and look out for pedestrians and obstacles on the road

Ultrasonic sensors may be used to measure the position of objects very close to the vehicle, such as curbs and other vehicles when parking

The information from all of the sensors is analysed by a **central computer** that manipulates the steering, accelerator and brakes. Its software must understand the rules of the road, both formal and informal

Radar sensors monitor the position of other vehicles nearby. Such sensors are already used in adaptive cruise-control systems

Source: *The Economist*

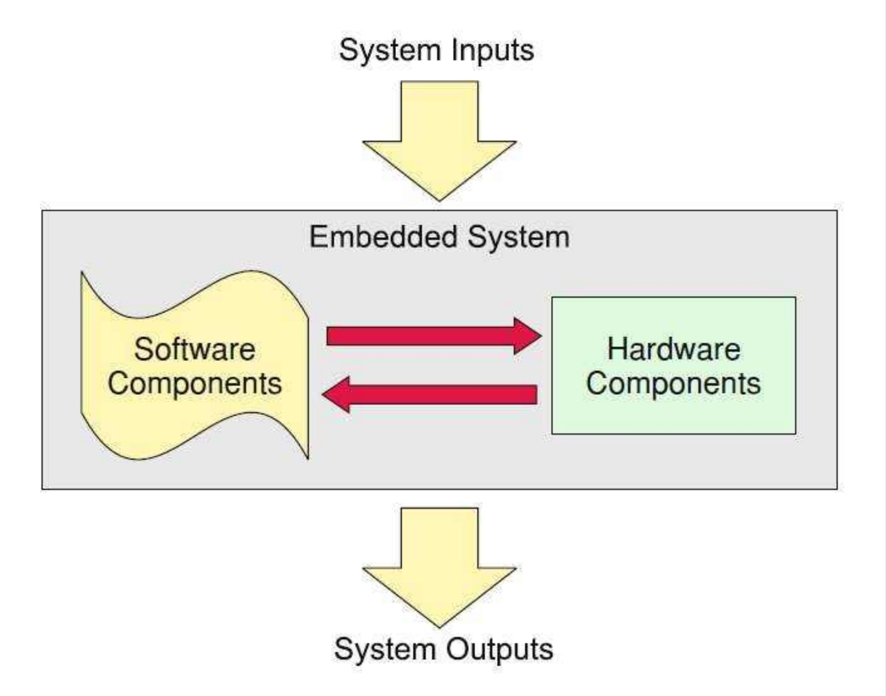
Introduction to Embedded Systems

- Some common characteristics of embedded systems
  - Perform a specific task or a small set of tasks
  - Reactive and real-time
    - Continually reacts to changes in the system’s environment
    - Many embedded systems must perform tasks in real-time
      - Soft real time
      - Hard real time
  - Tight constraints on
    - Cost
    - Energy
    - Size
    - Memory

Introduction to Embedded Systems

Typical inputs in an embedded system are process variables and parameters that arrive via sensors

Firmware is a computer program typically stored in a non-volatile memory embedded in a hardware device

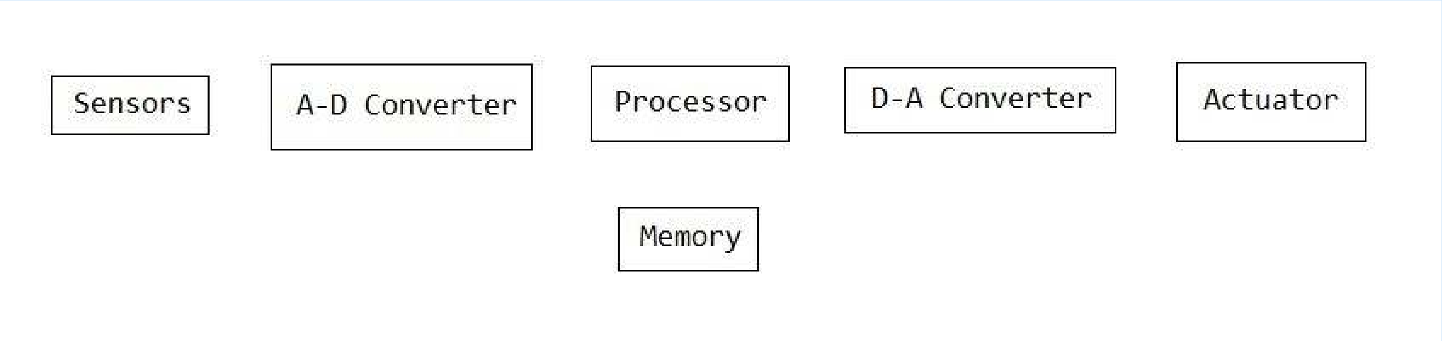


The outputs are in the form of control actions on system actuators or processed information for users or other subsystems within the application

General view of an embedded system

Introduction to Embedded Systems

- Basic hardware components
  - Sensor measures physical quantity
  - A-D converter converts analog signal sent by sensor into a digital signal
  - Processor processes the data
  - Memory stores data and instructions
  - D-A converter converts digital data fed by processor to analog data
  - Actuator responsible for moving and controlling a mechanism or system

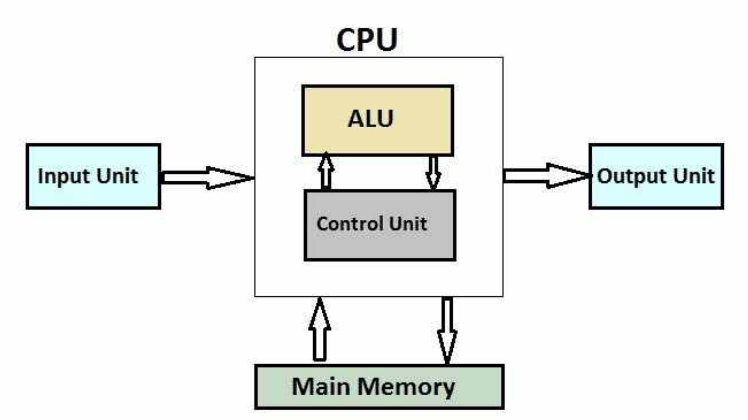


Introduction to Embedded Systems

- Software Components
  - Firmware is a computer program typically stored in a non-volatile memory embedded in a hardware device
  - Firmware is not meant to be modifiable by users, although some systems could provide means of performing upgrades

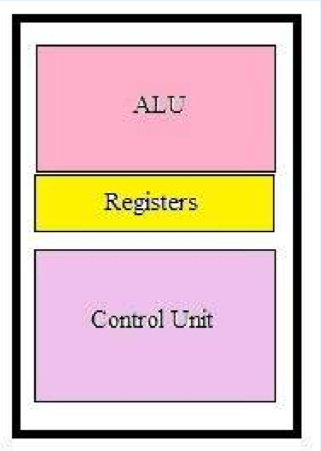
Introduction to Embedded Systems

- Basic operations of computer system
  - CPU carries out all computations
    - Arithmetic logic unit (ALU) performs arithmetic, comparison, and logical operations
    - Control unit directs and coordinates operations in computer
  - Main memory temporarily stores data and program instructions during processing
  - IO units provide interface with outside world
    - The outside world may not always be a human being, it may be an environment



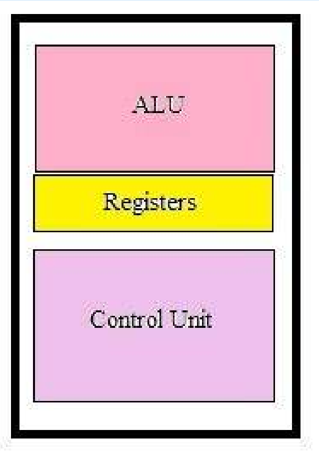
Introduction to Embedded Systems

- Processors are broadly classified into 3 major categories
  - General Purpose Microprocessors
  - Microcontrollers
  - Digital Signal Processors



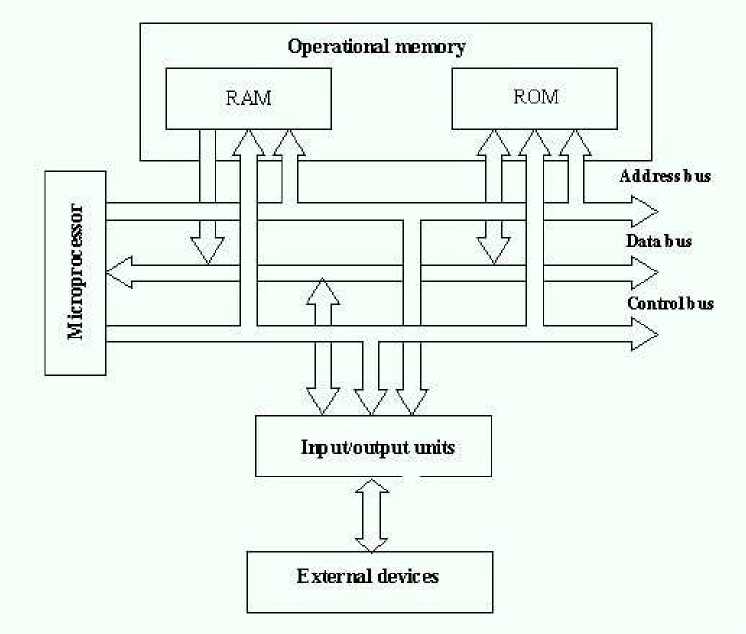
Introduction to Embedded Systems

- Microprocessor
  - It is basically entire CPU fabricated on a single chip
  - It consists of a set of registers to store temporary data
  - It consists of an ALU and CU
  - It consists of some mechanism to interface with external devices such as memory and IO through buses



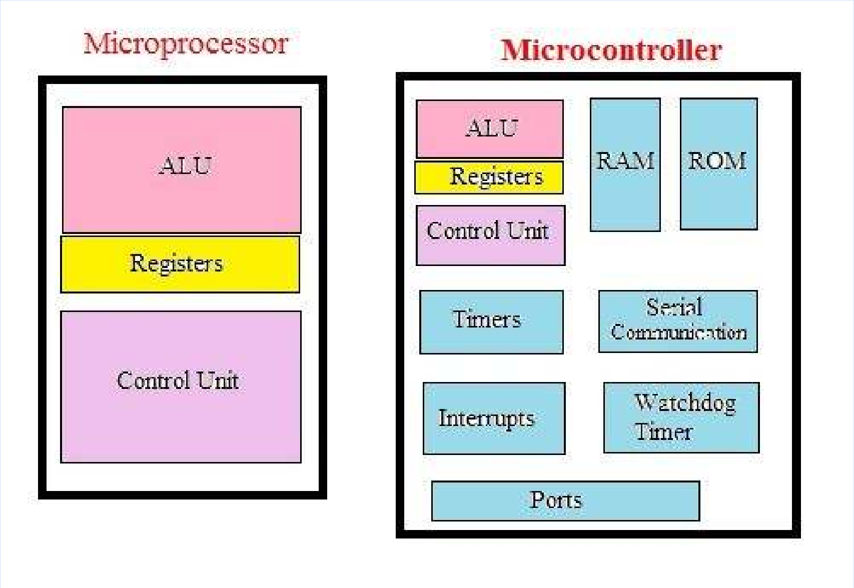
Introduction to Embedded Systems

- Microcomputer
  - It is a computer system build using microprocessor
  - Since microprocessor does not contain memory and IO, we have to interface these to build a microcomputer



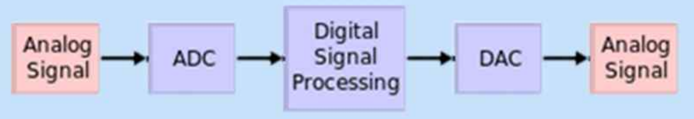
Introduction to Embedded Systems

- Microcontroller
  - A computer on a single chip



Introduction to Embedded Systems

- Digital Signal Processor
  - It is a specialized microprocessor chip, with its architecture optimized for the operational needs of digital signal processing
  - Digital signal processors are widely used in audio signal processing, telecommunications, digital image processing, radar, sonar and speech recognition systems, and in common consumer electronic devices such as mobile phones, disk drives and high-definition television (HDTV) products



A typical digital processing system

source: https://en.wikipedia.org/wiki/Digital_signal_processor

Introduction to Embedded Systems

- RISC Versus CISC Architectures
  - The instruction set architecture is the set of basic instructions that a processor understands
    - CISC (Complex Instruction Set Computing)
    - RISC (Reduced Instruction Set Computing)

- Complex Instruction Set Computer
  - The primary goal is to complete a task in as few lines of assembly as possible

```
MULT 2:3, 5:2
```

  - When executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and then stores the product in the appropriate register. Thus, the entire task of multiplying two numbers can be completed with one instruction
  - MULT is what is known as a *complex instruction*
  - CISC used in
    - Desktops
    - Laptops
    - servers

source: https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/

Introduction to Embedded Systems

- Reduced Instruction Set Computer
  - RISC processors only use simple instructions that can be executed within one clock cycle
  - The MULT command could be divided into three separate commands

```
LOAD A, 2:3
LOAD B, 5:2
PROD A, B
STORE 2:3, A
```

  - More RAM is needed to store the assembly level instructions
  - The compiler must also perform more work to convert a high-level language statement into code of this form
  - Typically used in microcontrollers that are used in embedded systems

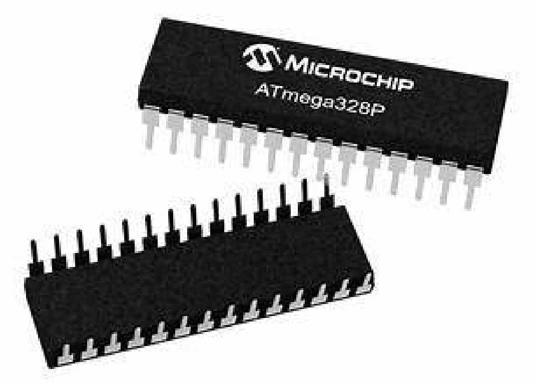
source: https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/

Open-Source Hardware for IoT

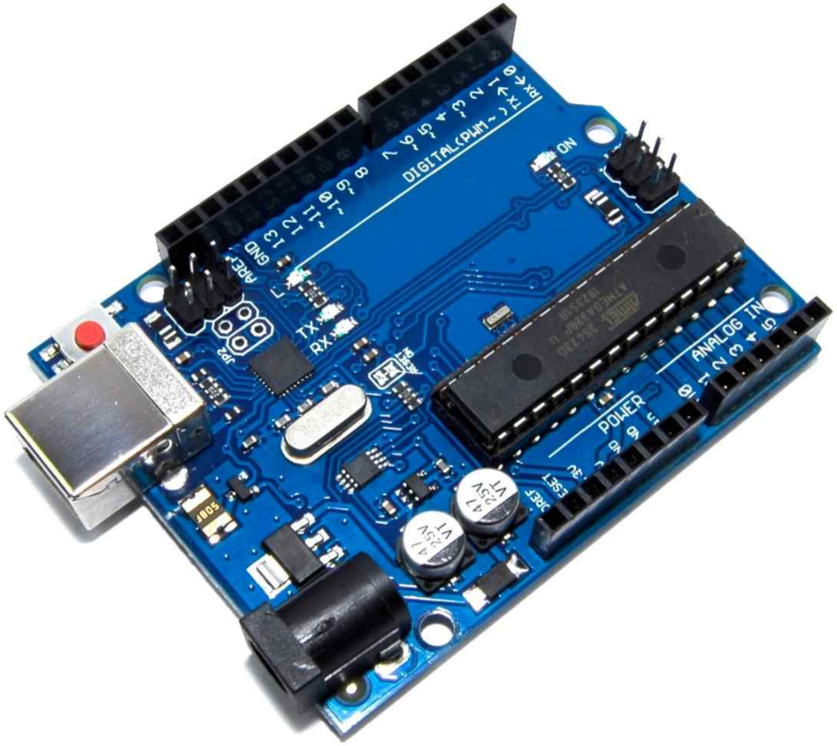
- Microcontrollers
- Arduino
- Raspberry Pi

Microcontroller

- **A microcontroller** is a compact integrated circuit designed to govern a specific operation in an embedded system.
- A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.
- Sometimes referred to as an embedded controller or microcontroller unit (MCU).
- Microcontrollers are found in:
  - vehicles,
  - robots,
  - medical devices,
  - mobile radio transceivers,
  - vending machines
  - home appliances,
  - among other devices.



Microcontroller chip



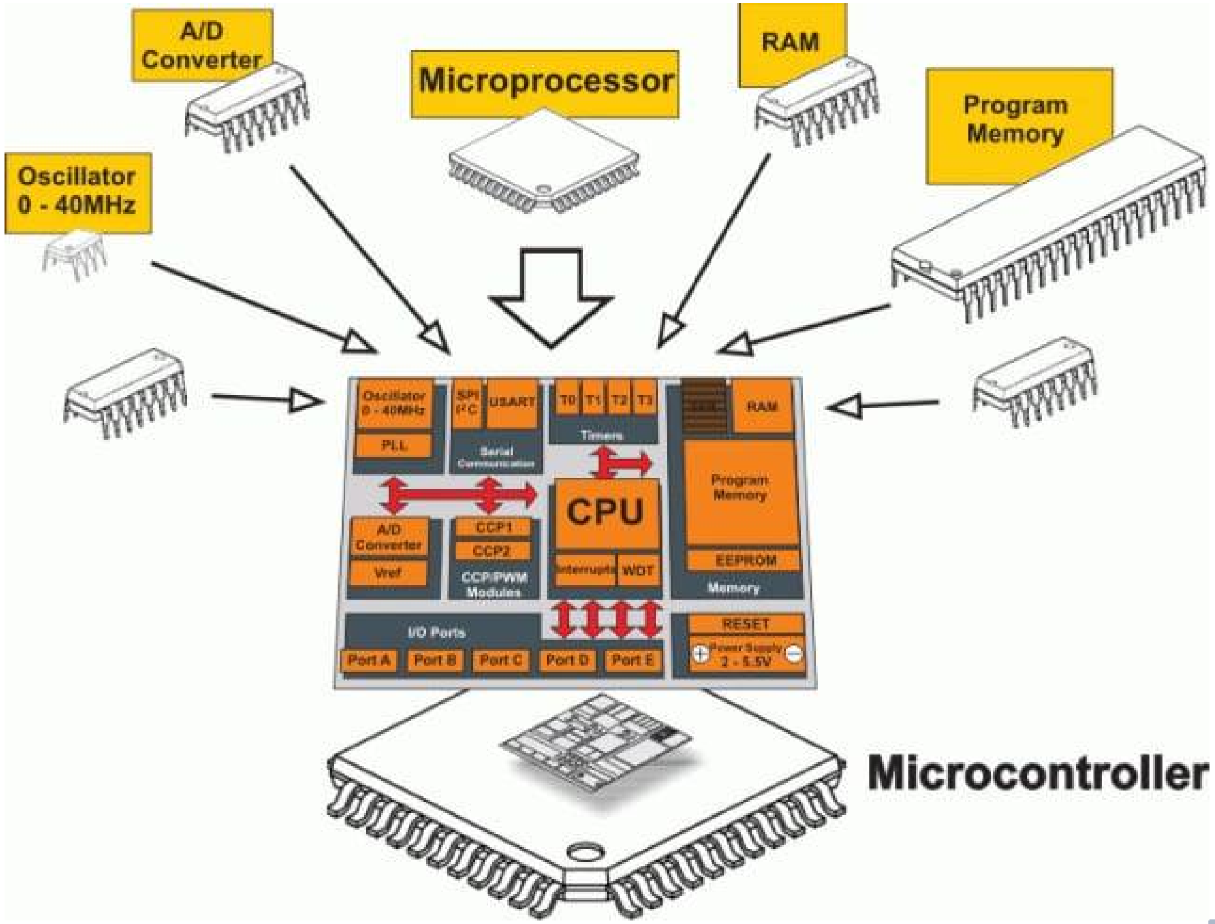
Microcontroller chip + board

https://www.techtarget.com/iotagenda/definition/microcontroller

Inside a Microcontroller: Essential Components

A microcontroller can be seen as a small computer, and this is because of the essential components inside of it;

- Central Processing Unit (CPU),
- Memory
  - Random-Access Memory (RAM),
  - Electrical Erasable Programmable Read-Only Memory (EEPROM).
- Flash Memory,
- Serial Bus Interface,
- Input/Output Ports (I/O Ports),

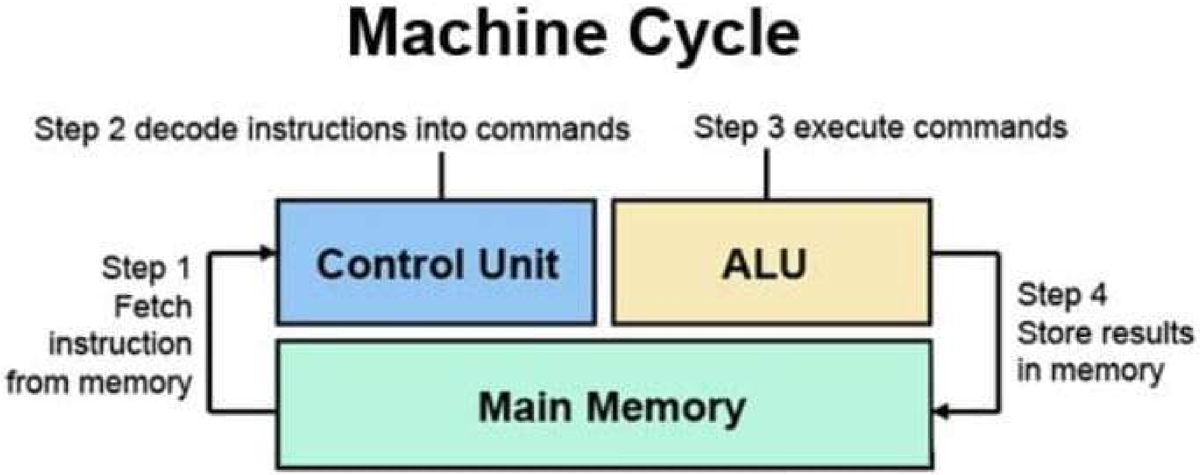


https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller

Inside a Microcontroller: Essential Components

Design of Microcontroller CPU

- Processing all the data input it receives and executes the required instructions.
- ALU performs arithmetic and logical operations,
- Control Unit (CU), which handles all of the processor's instruction executions.



Machine Cycle

Step 2 decode instructions into commands

Step 3 execute commands

Step 1 Fetch instruction from memory

Control Unit

ALU

Step 4 Store results in memory

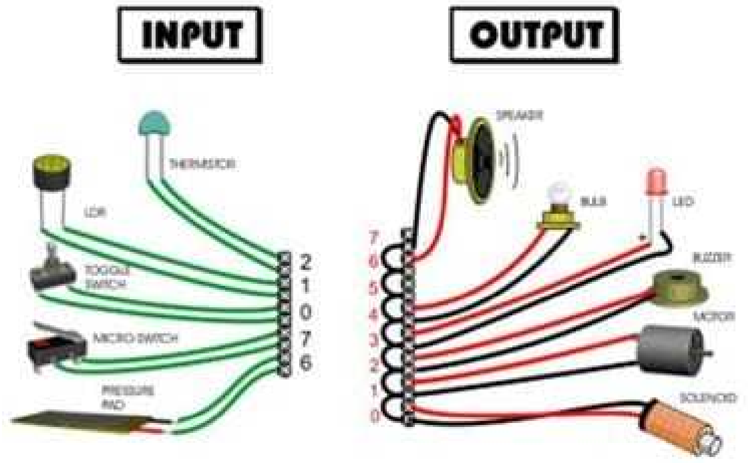
Main Memory

Microcontroller I/O Ports

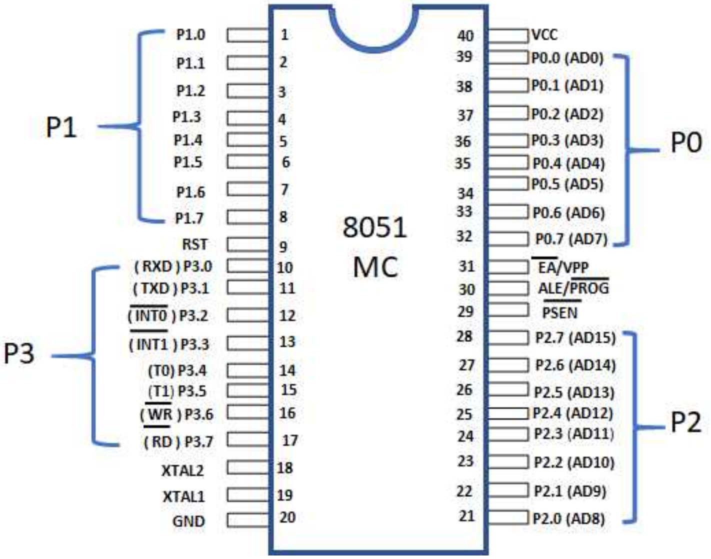
- I/O ports are what the microcontroller uses to connect to real-world applications.

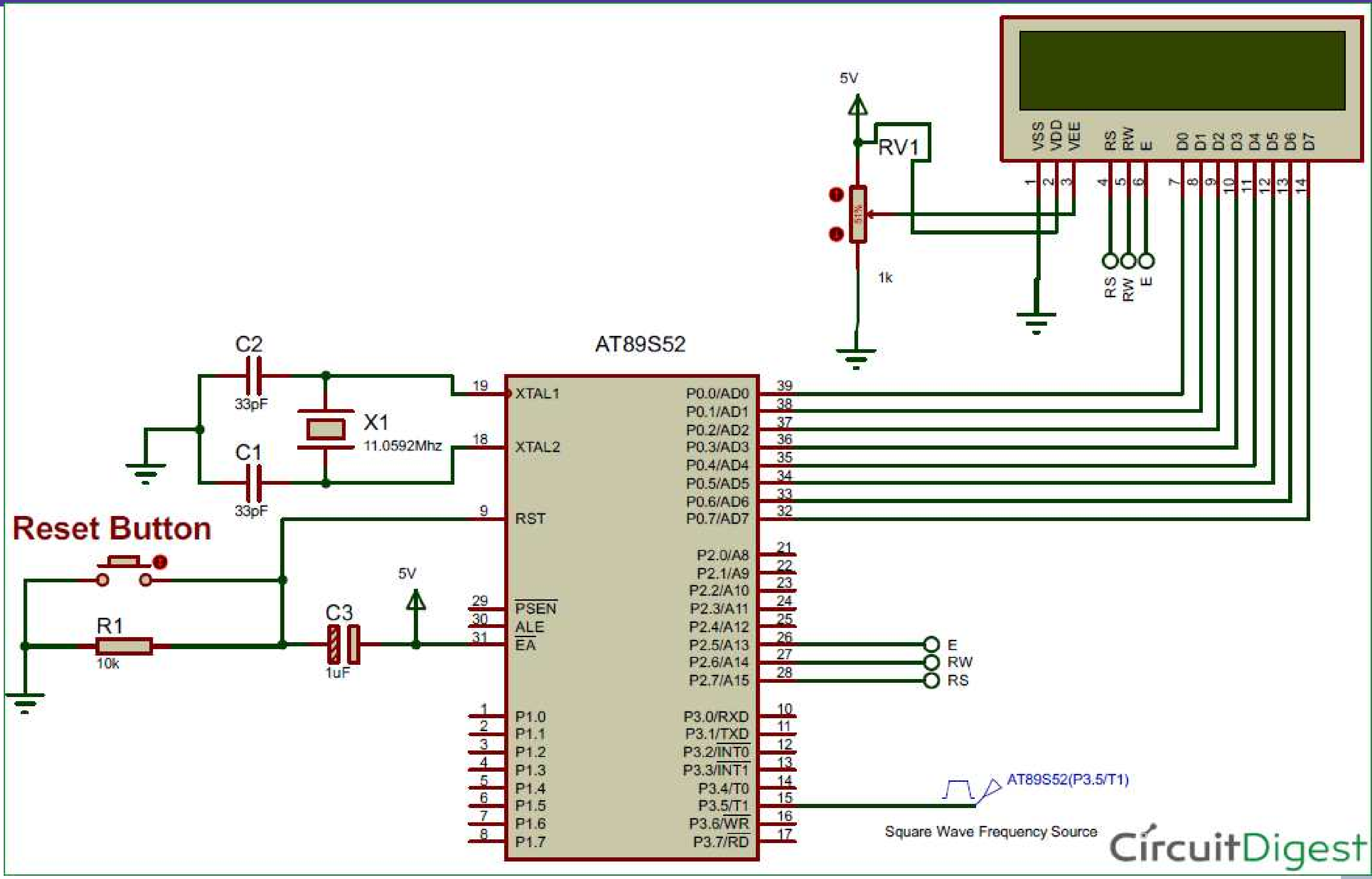
- Inputs such as temperature sensing, motion sensing, push buttons,…...

- Output ports such as LED lights, LCD, running a motor, speaker, …..

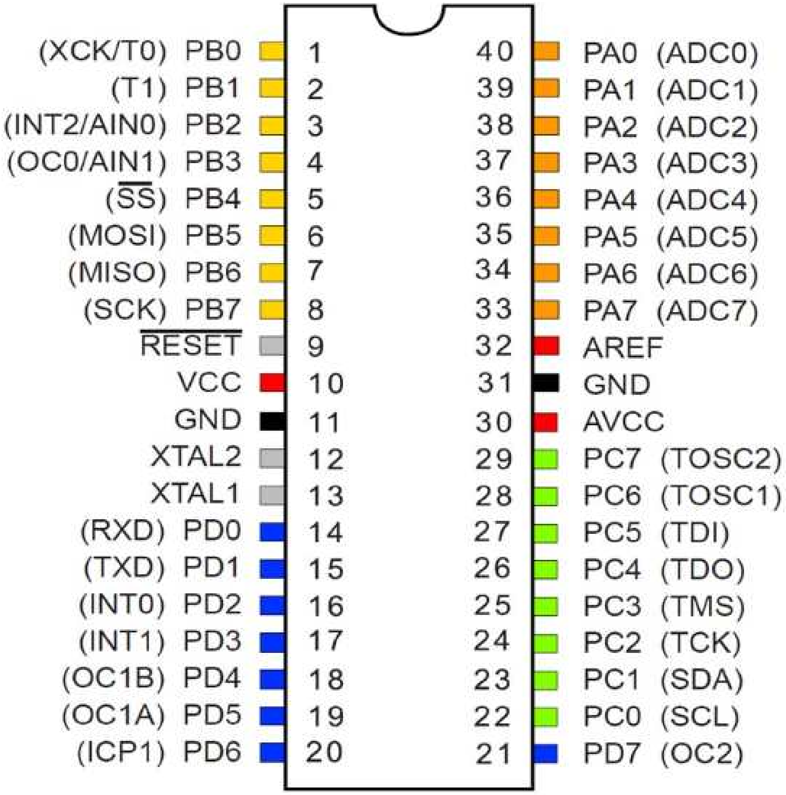


Simulation of the Microcontroller

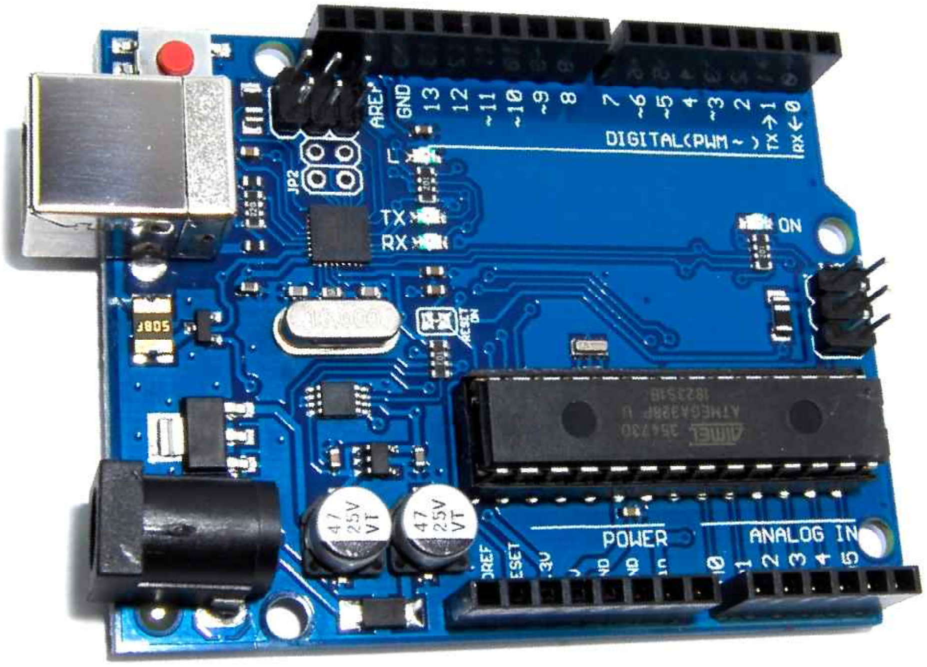




Types of the Microcontroller

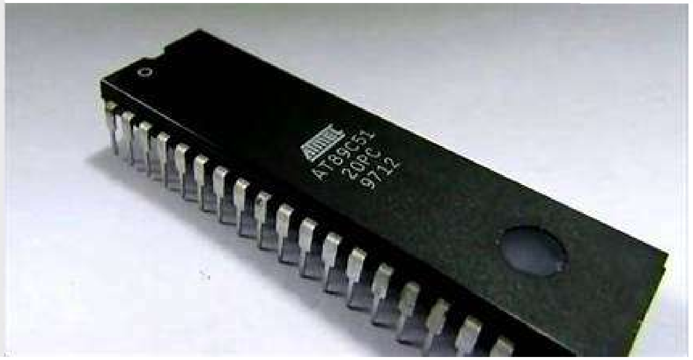


ATmega32A microcontroller which is 8-bit and 40 pin AVR chip.



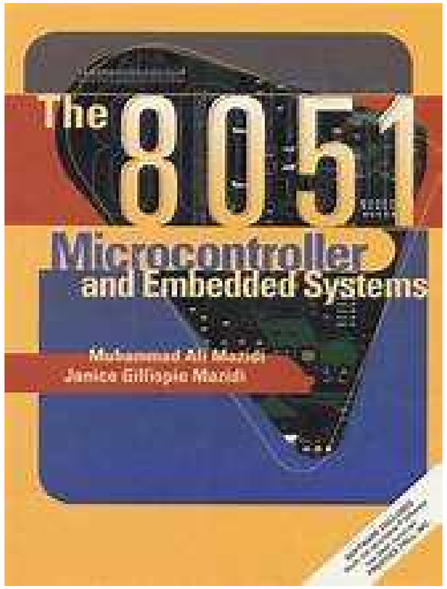
ATmega32A microcontroller Board.

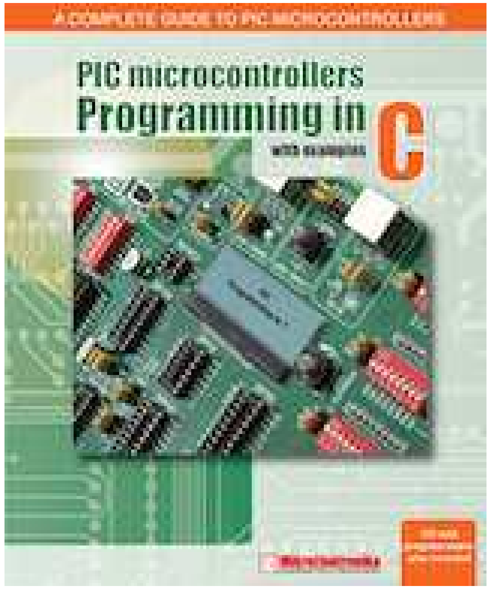
Types of the Microcontroller (SW/HD)

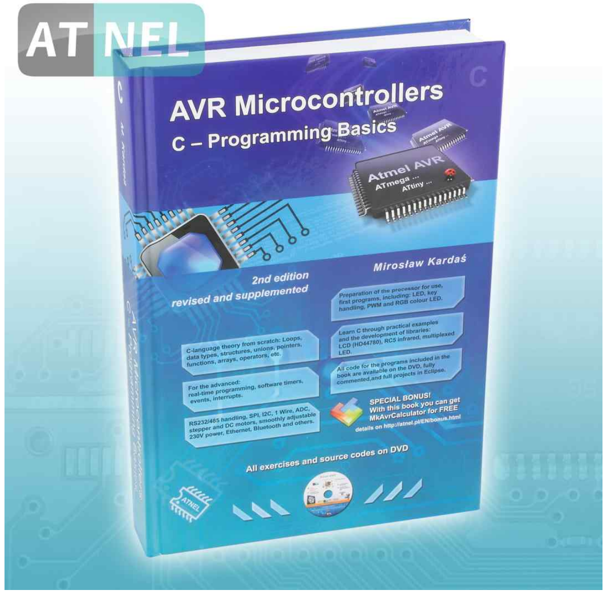


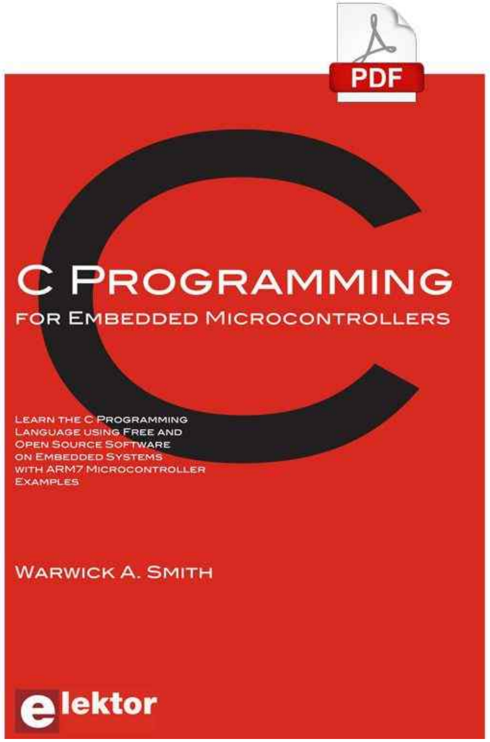
Language	Ease of Use	Performance	Memory Usage	Best For	Difficulty
C / C++	Moderate	High	Low	Complex embedded systems	Moderate to Difficult
Assembly	Difficult	Very High	Very Low	Performance-critical tasks	Very Difficult
Python (MicroPython)	Easy	Moderate	High	Prototyping, IoT, education	Easy to Moderate

References


The 8051
Microcontroller and Embedded Systems
Muhammad Ali Mazidi
Janice Gillispie Mazidi


A COMPLETE GUIDE TO PIC MICROCONTROLLERS
PIC microcontrollers
Programming in C
with examples
mikroElektronika


ATNEL
AVR Microcontrollers
C – Programming Basics
Atmel AVR
ATmega ...
ATtiny ...
Mirosław Kardaś
2nd edition
revised and supplemented
SPECIAL BONUS!
All exercises and source codes on DVD


PDF
C Programming
for Embedded Microcontrollers
Learn the C Programming
Language using Free and
Open Source Software
on Embedded Systems
with ARM7 Microcontroller
Examples
Warwick A. Smith
elektor

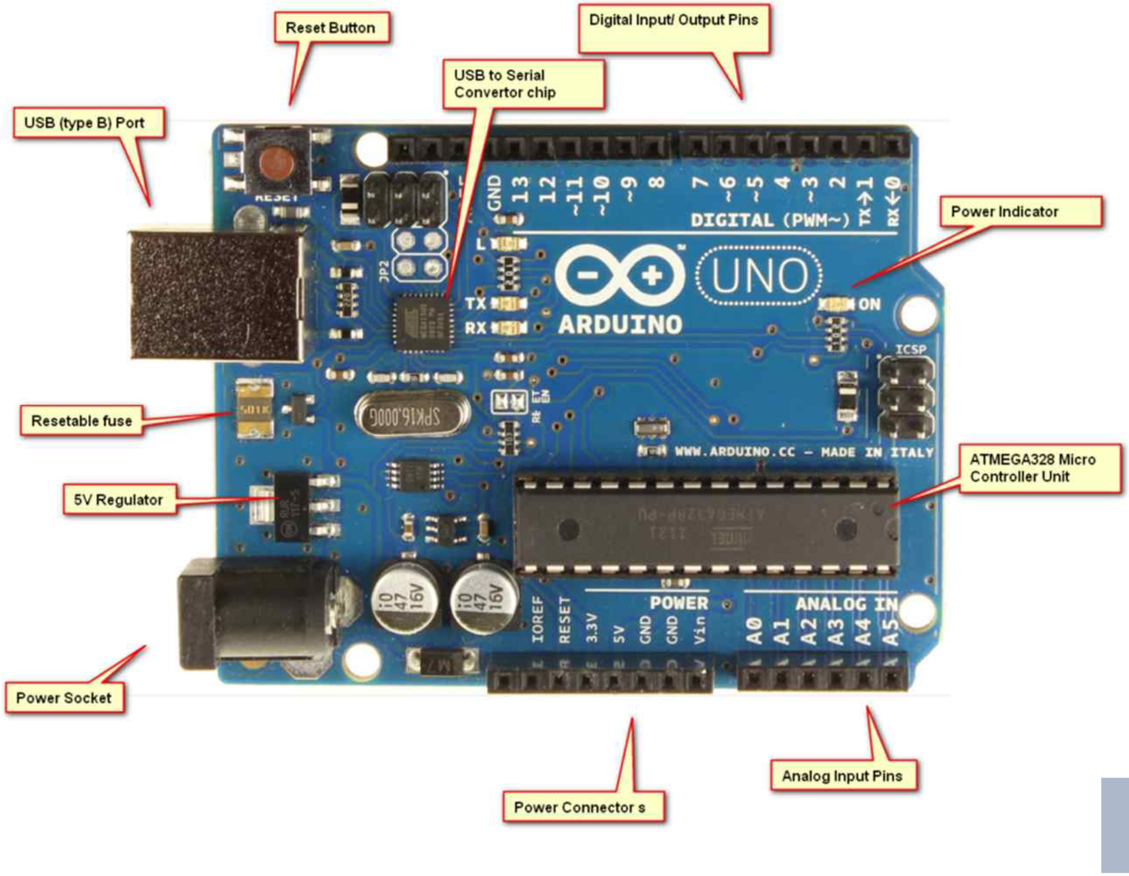
Arduino

Arduino

- 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 – and turn it into an output – activating a motor, turning on an LED, publishing something online.

- The Arduino project started in 2005 in Italy to make a low-cost and simple solution to create digital projects.



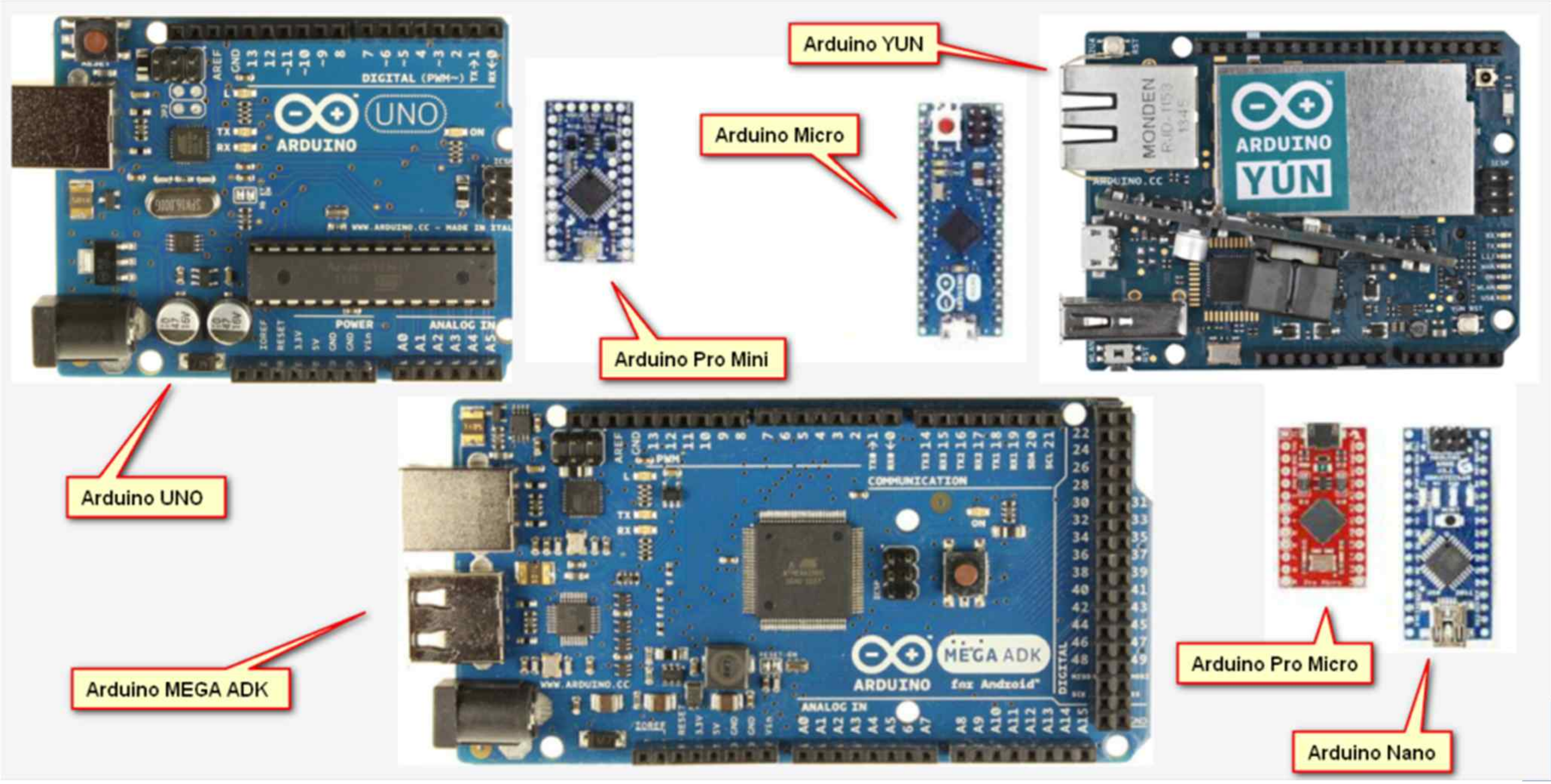
Arduino Hardware

The Arduino UNO, MEGA and ZERO are the best.

The Arduino UNO R3 is

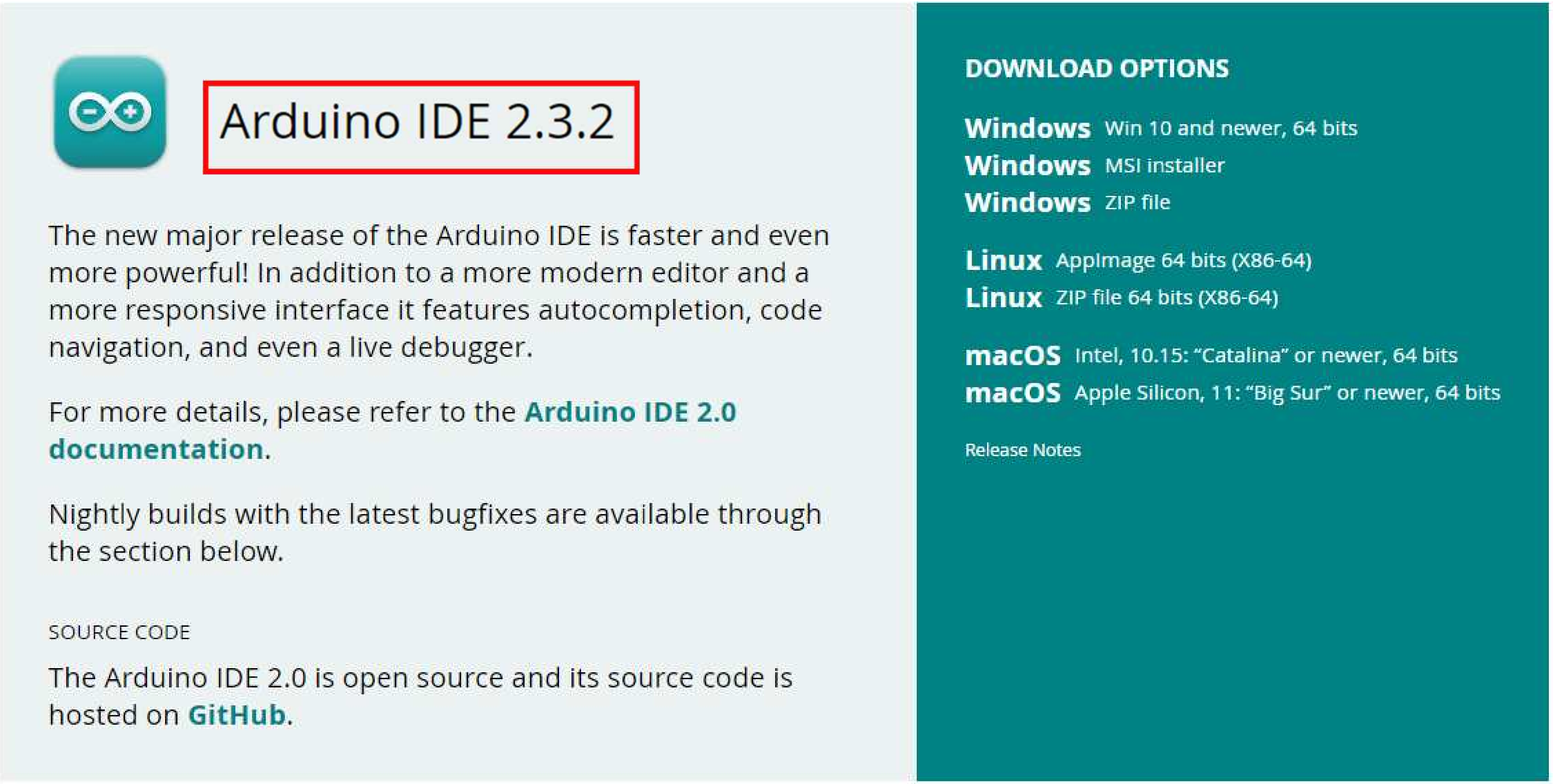
- very easy to use,
- USB type-B port to connect with Computer
- Power socket
- fairly cheap.

It is compatible with most projects and code examples you will find on the internet.



Arduino IDE software

As of January 4, 2017, ARDUINO 1.8.0 is the latest version of Arduino IDE.



Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

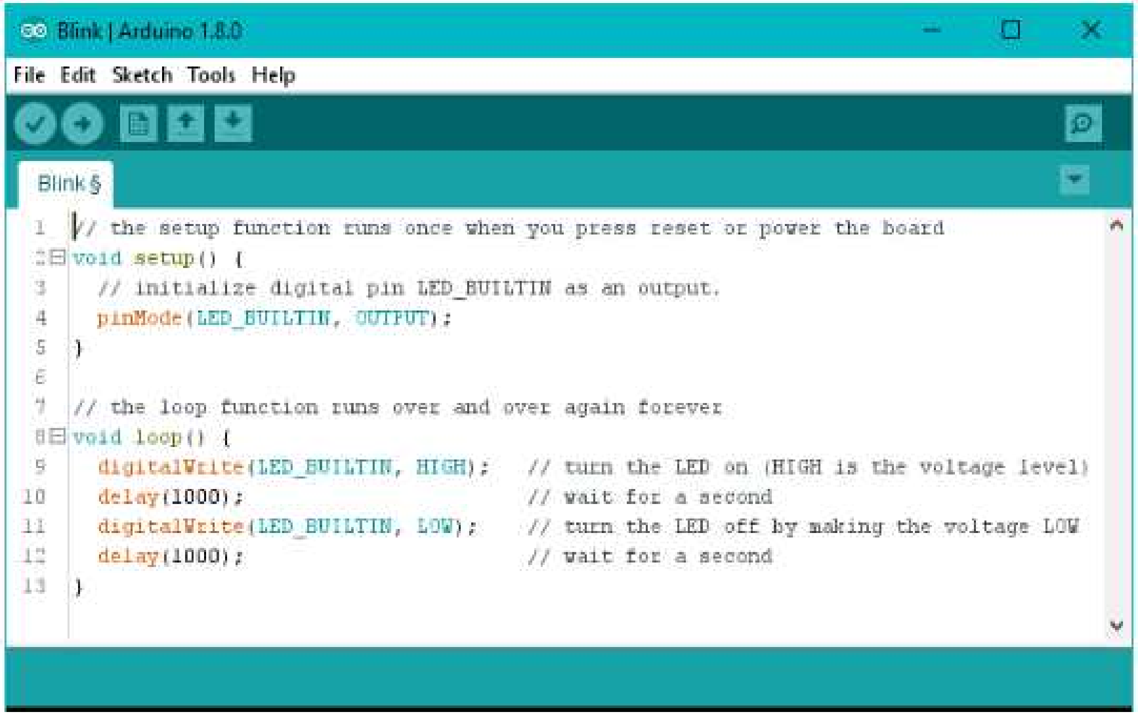
macOS Intel, 10.15: "Catalina" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

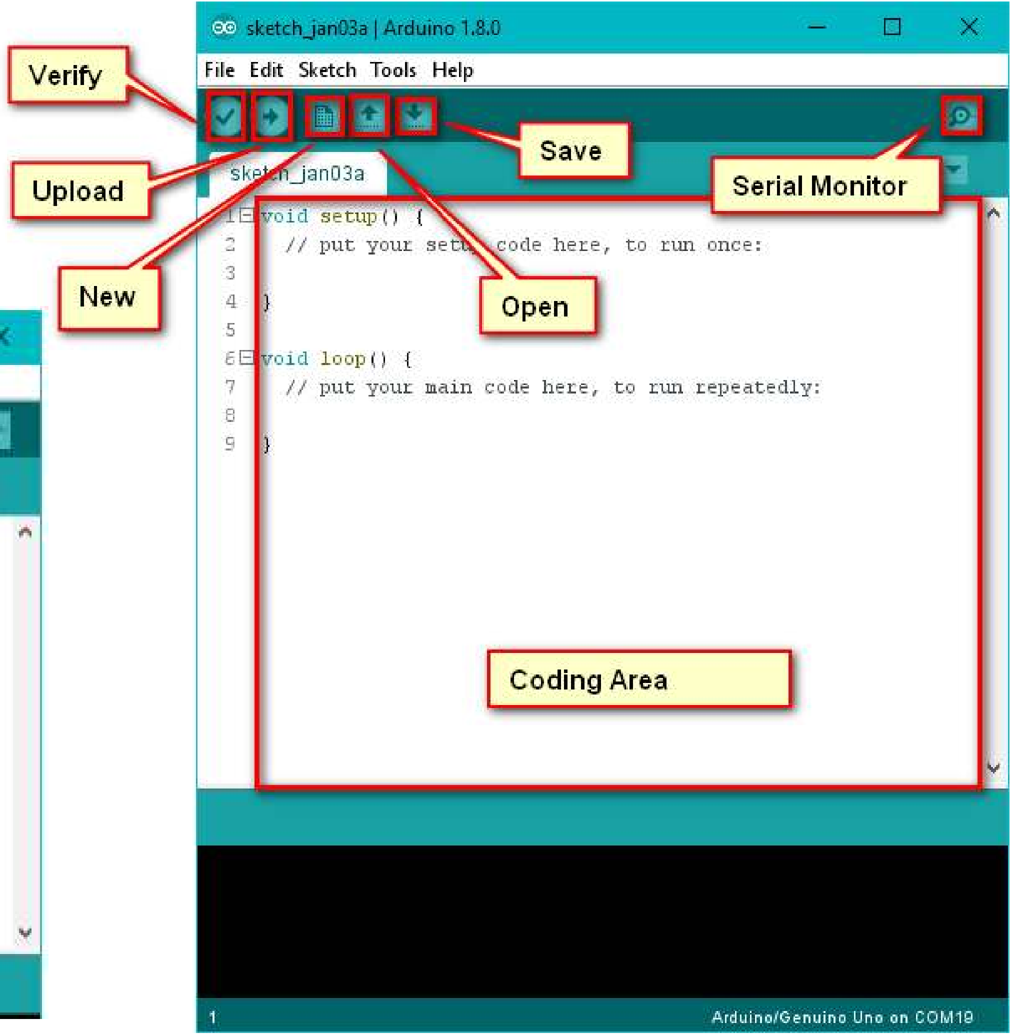
https://www.arduino.cc/en/software

Arduino IDE software

- After downloading and installing the software, you can open and run the IDE.
- You will see the following window on your computer.



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

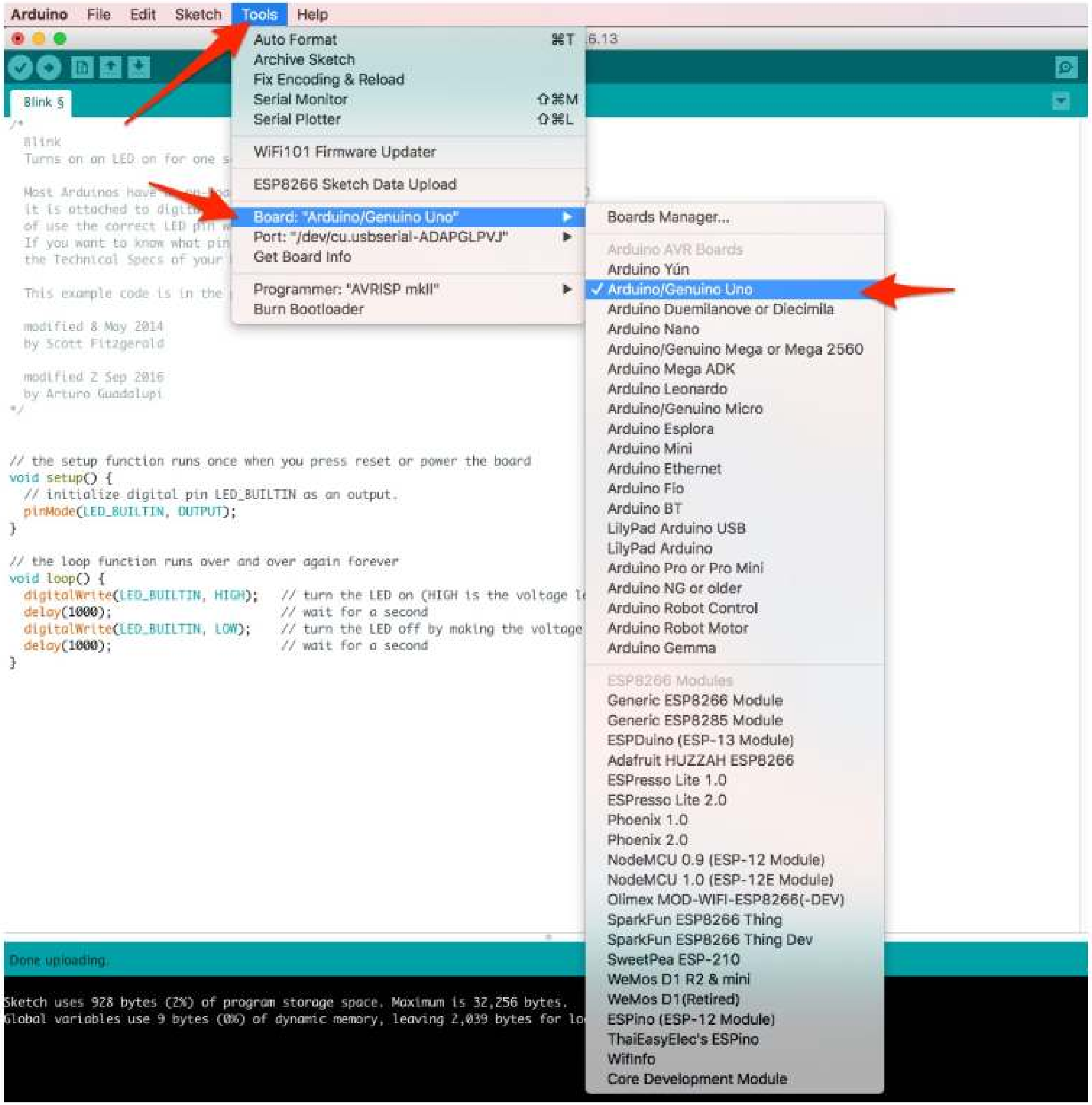


sketch_jan03a | Arduino 1.8.0
File Edit Sketch Tools Help
Verify
Upload
New
Open
Save
Serial Monitor
sketch_jan03a
Coding Area
1
Arduino/Genuino Uno on COM19

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino IDE software

- Before we can upload the program, we need to get our Arduino board and Port configured in the IDE.
- First, Select **Tools** -> **Board** and click on the **Arduino/Genuino Uno**



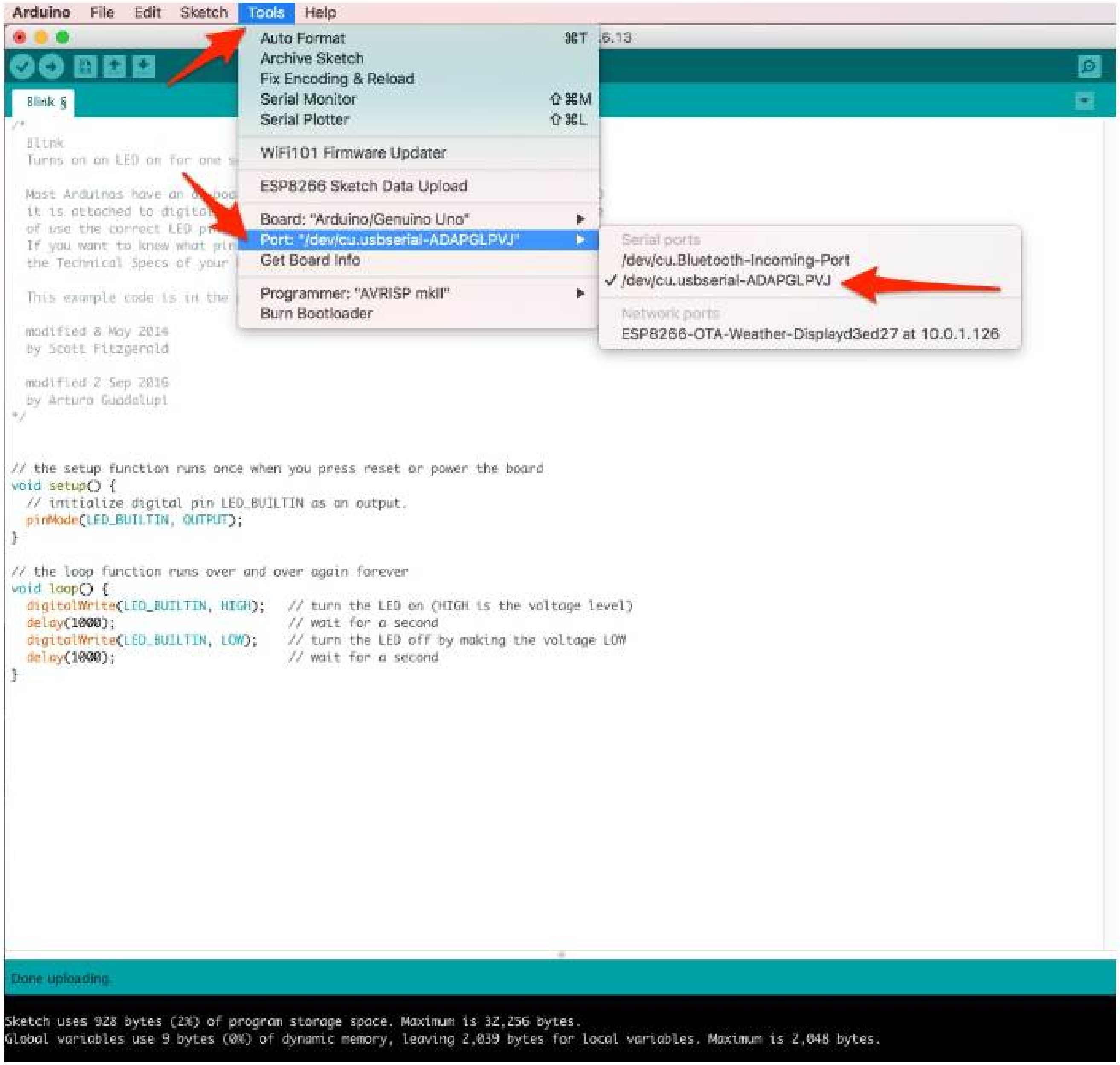
```
/*
  Blink
  Turns on an LED on for one s

  Most Arduinos have
  it is attached to digit
  of use the correct LED pin
  If you want to know what pin
  the Technical Specs of your

  This example code is in the

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage le
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage
  delay(1000);                       // wait for a second
}
```

Done uploading.

```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32,256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2,039 bytes for lo
```

Arduino IDE software

Next, select the proper USB Port which will usual contain the words “usbserial” depending on your Operating System.



Arduino
File
Edit
Sketch
Tools
Help
Blink §
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 Firmware Updater
ESP8266 Sketch Data Upload
Board: "Arduino/Genuino Uno"
Port: "/dev/cu.usbserial-ADAPGLPVJ"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
/dev/cu.Bluetooth-Incoming-Port
✓ /dev/cu.usbserial-ADAPGLPVJ
Network ports
ESP8266-OTA-Weather-Displayd3ed27 at 10.0.1.126

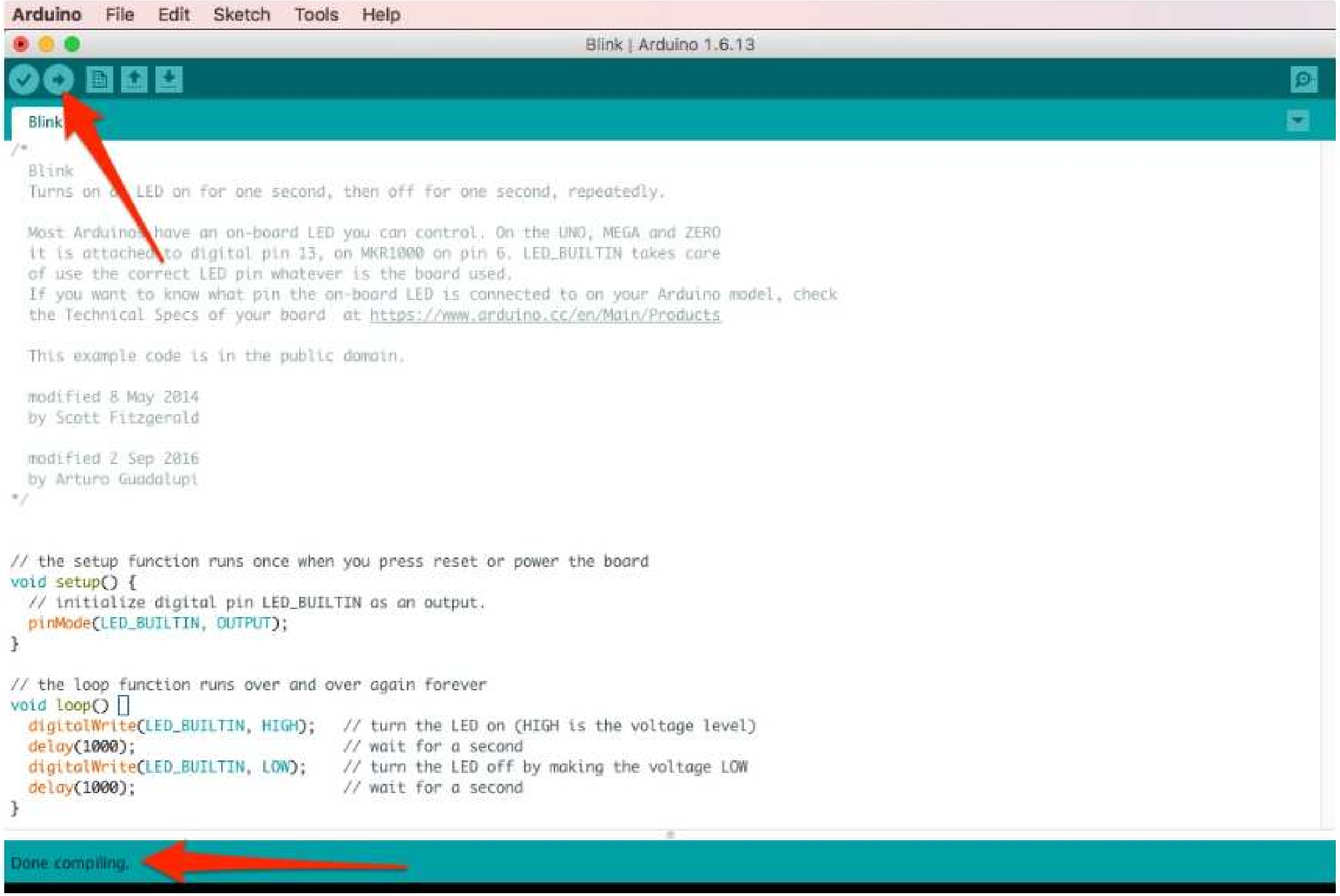
```
/*
  Blink
  Turns on an LED on for one s

  Most Arduinos have an on-boa
  it is attached to digital
  of use the correct LED pi
  If you want to know what pin
  the Technical Specs of your

  This example code is in the

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done uploading.

```
Sketch uses 928 bytes (2%) of program storage space. Maximum is 32,256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2,039 bytes for local variables. Maximum is 2,048 bytes.
```

Arduino IDE software

Now you are ready to hit the upload button! This is commonly referred to as “Uploading the Sketch”



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

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN takes care
  of use the correct LED pin whatever is the board used.
  If you want to know what pin the on-board LED is connected to on your Arduino model, check
  the Technical Specs of your board  at https://www.arduino.cc/en/Main/Products

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi
*/


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done compiling.

Any Questions!

Raspberry Pi 3B+ Pin Layout

□BCM Pin Numbering (Broadcom GPIO numbers)

This refers to the GPIO pins based on the Broadcom SoC (System on Chip) numbering scheme. Each pin is identified by its Broadcom GPIO number.

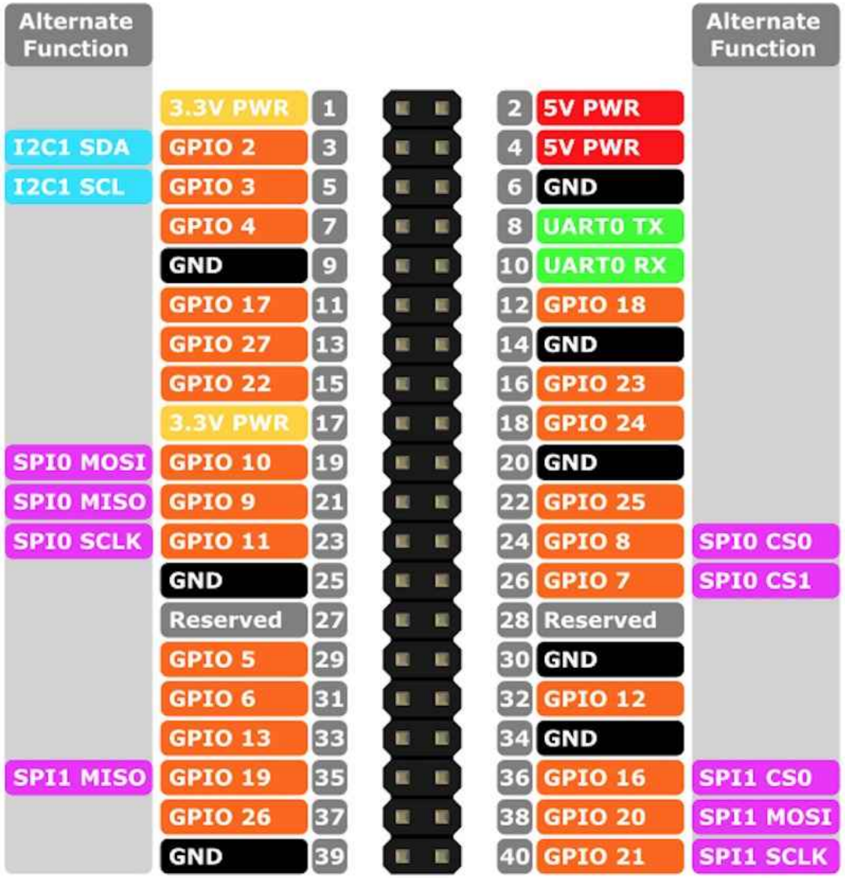
□WiringPi Pin Numbering

This is a different numbering system used by the WiringPi library.

WiringPi provides a simple interface for controlling the GPIO pins, and it uses its own numbering scheme, which may not match the BCM numbering or the physical pin numbering.

□Physical Pin Numbering

This refers to the actual physical layout of the pins on the Raspberry Pi's GPIO header. It starts from pin 1 (top left corner) and counts across each row of pins. Pin numbers in this scheme are sequential, from 1 to 40. analog ADC



Alternate Function					Alternate Function
	3.3V PWR	1	2	5V PWR	
I2C1 SDA	GPIO 2	3	4	5V PWR	
I2C1 SCL	GPIO 3	5	6	GND	
	GPIO 4	7	8	UART0 TX	
	GND	9	10	UART0 RX	
	GPIO 17	11	12	GPIO 18	
	GPIO 27	13	14	GND	
	GPIO 22	15	16	GPIO 23	
	3.3V PWR	17	18	GPIO 24	
SPI0 MOSI	GPIO 10	19	20	GND	
SPI0 MISO	GPIO 9	21	22	GPIO 25	
SPI0 SCLK	GPIO 11	23	24	GPIO 8	SPI0 CS0
	GND	25	26	GPIO 7	SPI0 CS1
	Reserved	27	28	Reserved	
	GPIO 5	29	30	GND	
	GPIO 6	31	32	GPIO 12	
	GPIO 13	33	34	GND	
SPI1 MISO	GPIO 19	35	36	GPIO 16	SPI1 CS0
	GPIO 26	37	38	GPIO 20	SPI1 MOSI
	GND	39	40	GPIO 21	SPI1 SCLK

https://pinout.xyz/pinout/3v3_power

Raspberry Pi 3B+ Pin Layout

❑**BCM Pi**

This re

(Syste

its Bro

❑**WiringP**

This is

library

Wiring

pins, a

match

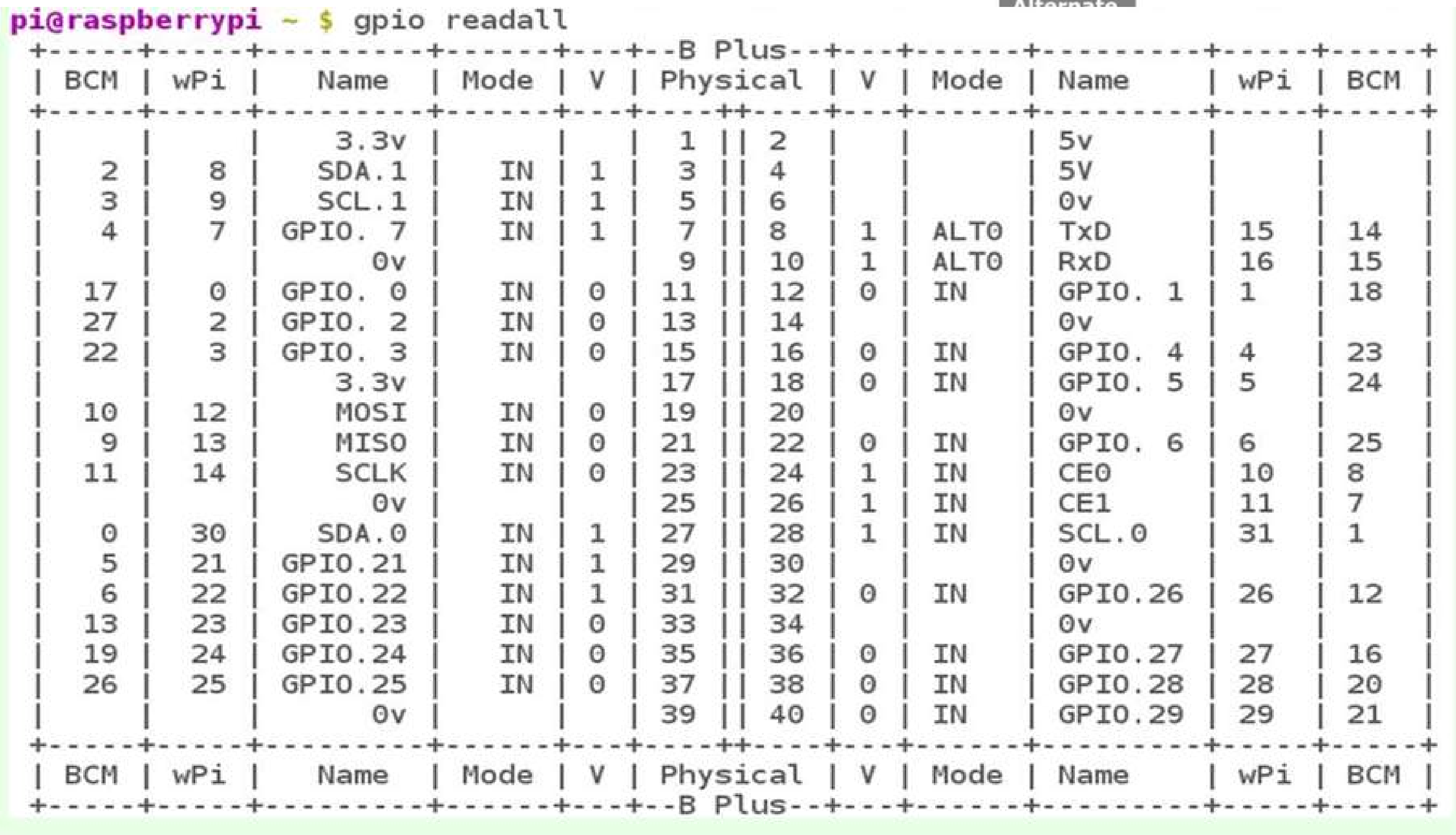
❑**Physical**

This re

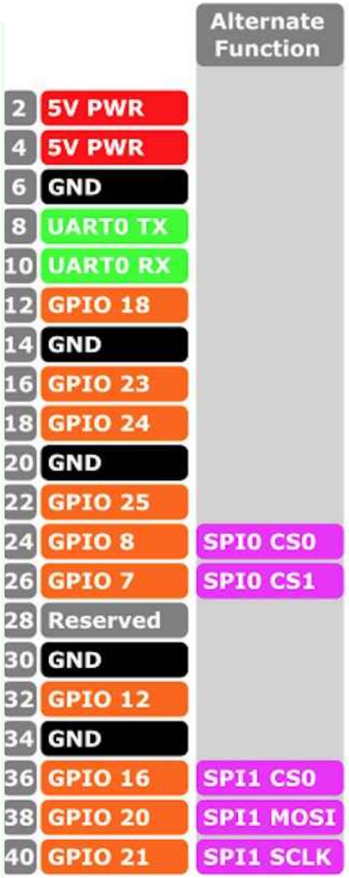
Raspb

corner

this sc



```
pi@raspberrypi ~ $ gpio readall
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
```



		Alternate Function
2	5V PWR	
4	5V PWR	
6	GND	
8	UART0 TX	
10	UART0 RX	
12	GPIO 18	
14	GND	
16	GPIO 23	
18	GPIO 24	
20	GND	
22	GPIO 25	
24	GPIO 8	SPI0 CS0
26	GPIO 7	SPI0 CS1
28	Reserved	
30	GND	
32	GPIO 12	
34	GND	
36	GPIO 16	SPI1 CS0
38	GPIO 20	SPI1 MOSI
40	GPIO 21	SPI1 SCLK

https://pinout.xyz/pinout/3v3_power