Computer Peripherals

CSE 315: Peripheral & Interfacing

Discussion Materials:

- Storage Devices
  - Hard Disk Drive
  - Flash Memory (PD & MC)
- I/O Devices
  - Displays (LCD, LED)
  - Printers (DMP, IJP & LP)
  - Scanners
  - Pointing Devices

Peripheral

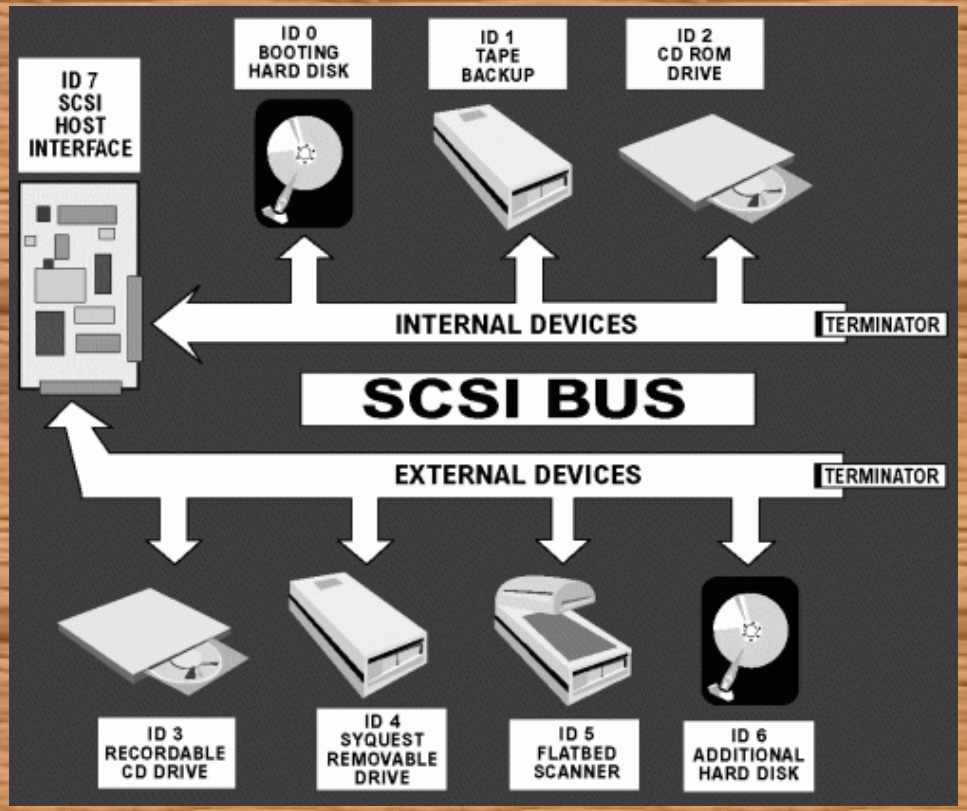
- Devices that are ***external*** to the main processing function of the computer
  - Other than the CPU, memory, power supply
- Classified as ***input***, ***output***, and ***storage***
- Connected via
  - Ports like
    - Serial, Parallel, USB etc
  - Interface to systems bus like
    - Small Computer System Interface (SCSI), Integrated Drive Electronics (IDE), Personal Computer Memory Card International Association (PCMCIA)

SCSI (Small Computer System Interface)

- **Small Computer System Interface** (**SCSI)** is a set of standards for physically connecting and transferring data between computers and peripheral devices. The SCSI standards define commands, protocols, electrical, optical and logical interfaces. SCSI is most commonly used for hard disk drives and tape drives, but it can connect a wide range of other devices, including scanners and CD drives, although not all controllers can handle all devices. The SCSI standard defines command sets for specific peripheral device types; the presence of "unknown" as one of these types means that in theory it can be used as an interface to almost any device, but the standard is highly pragmatic and addressed toward commercial requirements.

SCSI (contd.)


© 2006 HowStuffWorks

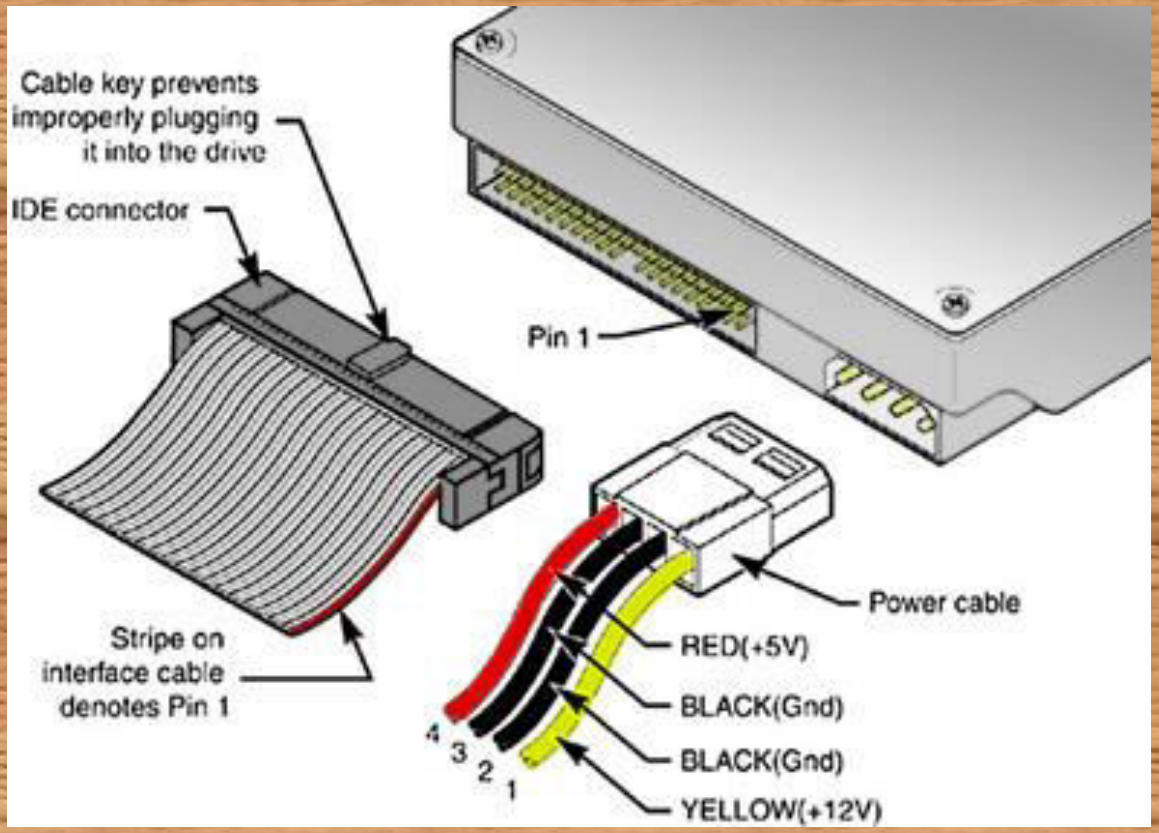

ID 7
SCSI
HOST
INTERFACE
ID 0
BOOTING
HARD DISK
ID 1
TAPE
BACKUP
ID 2
CD ROM
DRIVE
INTERNAL DEVICES
TERMINATOR
SCSI BUS
EXTERNAL DEVICES
TERMINATOR
ID 3
RECORDABLE
CD DRIVE
ID 4
SYQUEST
REMOVABLE
DRIVE
ID 5
FLATBED
SCANNER
ID 6
ADDITIONAL
HARD DISK

IDE (Integrated Drive Electronics)

- Nominally called the IDE (Integrated Drive Electronics) bus; how ever it's more correctly known as the ATA (Advanced Technology Attachment) specification [ATA Bus]. The IDE bus is used in Personal Computers [PCs] as a hard-drive or peripheral bus to interconnect the PC mother board and a hard drive. The IDE bus is a Parallel bus. With the introduction of the Serial ATA [SATA] specification Parallel ATA [IDE] is now being called PATA.

- The specification has been up-graded a number of times each building on the past specification. ATA-1 and 2 were single documents, but like SCSI, after ATA-2 the specification was divided into a number of different documents. Most maintain backward compatibility, keeping in mind the cable changed. Each new version of the standard produced an increase in bus speed. The data transfer rate is shown after each version listed below. The maximum IDE bus speed is 133MBytes/sec [133MBps].

IDE (contd.)


Cable key prevents improperly plugging it into the drive
IDE connector
Pin 1
Stripe on interface cable denotes Pin 1
Power cable
RED(+5V)
BLACK(Gnd)
BLACK(Gnd)
YELLOW(+12V)
4
3
2
1

PCMCIA (Personal Computer Memory Card International Association)

- In computing, **PC Card** is a configuration for computer parallel communication peripheral interface, designed for laptop computers. Originally introduced as **PCMCIA**, the PC Card standard as well as its successors like CardBus were defined and developed by the *Personal Computer Memory Card International Association* (PCMCIA).

PCMCIA (contd.)

Interfacing

- An interface is the point of interaction with software, or computer hardware, or with peripheral devices

There are two types of interfaces:

- 1. Hardware Interfaces.
- 2. Software Interfaces.

Interfacing Contd.

- HARDWARE INTERFACES: Hardware interfaces exist in computing systems between many of the components such as the various buses, storage devices, other I/O devices, etc.

- SOFTWARE INTERFACES: A software interface may refer to a range of different types of interface at different levels.

Interfacing Contd.

Different levels of software interfacing.

- An operating system may interface with pieces of hardware.
- Applications or programs running on the operating system may need to interact via streams
- In object oriented programs, objects within an application may need to interact via methods.

Ports

- A port serves as an interface between the computer and other computers or peripheral devices

Primarily there are two types of ports:

1. Physical Ports
2. Virtual Ports

Physical Ports:

- Physical ports are used for connecting a computer trough a cable and a socket to a peripheral device.

Physical computer ports list includes:

a. Serial ports (DB9 socket)

b. USB ports (USB 2.0 or 3.0 socket / connector)

c. Parallel ports (DB25 socket / connector

Virtual Ports

- Virtual ports are data gates that allow software application (network) to use hardware resources without any interference.

This computer ports (network ports) are defined by IANA (Internet Assigned Numbers Authority).

Used by TCP (Transmission Control Protocol), UDP (User Datagram Protocol), DCCP (Datagram Congestion Control Protocol) and SCTP (Stream Control Transmission Protocol)

Storage Devices: Terminology

- Medium
  - The technology or product type that holds the data
- Access Time
  - The time to locate data and read it
  - Calculated as an average in seconds e.g. - s, ms, µs, ns, etc
- Transfer Rate
  - Amount of data moves per second
  - Calculated in bytes/second e.g. – KBPS, MBPS etc

Storage Devices: Primary

- Primary memory
  - Named as cache or conventional memory
  - Has an immediate access by the CPU
- Expanded storage
  - RAM – Random Access Memory
  - A buffer between cache memory and secondary memory

Storage Devices: Secondary

- Secondary storage
  - Permanent memory
  - Non volatile memory
  - Data and programs must be copied to primary memory for CPU access
  - Electro-Mechanical devices
  - Direct access storage devices (DASDs)
  - Online storage
  - Offline storage – loaded when needed

Storage Devices: Hierarchy

	Devices	Access Time	Transfer rate
Primary storage	CPU registers	-	-
	Cache memory	15-30 ns	-
	Conventional memory	50-100 ns	-
	Expanded memory	75-500 ns	-
Secondary storage	Hard disk	10-50 ms	600-6000 Kbytes/s
	Floppy disk	95 ms	100-200 Kbytes/s
	CD-ROM	100-600 ms	150-1000 Kbytes/s
	Tape	0.5+ s	200-3000 Kbytes/s

Storage Devices: Technology

- Primary storage
  - Semiconductor technology e.g.- RAM
  - Volatile i.e. contents are depended on power
- Secondary storage
  - Magnetic technology e.g. Hard Disk Drives
  - Non-volatile i.e. contents are not depended on power

Storage Devices: Intervention

- Online storage
  - Storage that is accessible to programs without human intervention
  - ***Primary*** & ***Secondary*** storages are ***online***
  - Example- RAM, HDD etc
- Offline storage
  - Storage that is not accessible to programs without human intervention
  - Sometimes called ***archival storage***
  - Example- CD, DVD, External HDD etc

Assignment for next class:

- Serial port, Parallel port, USB 10
- Relation between peripheral and interfacing 10

Due date: Next lecture day

Thank you

Intro to Arduino and different types of Arduino boards

CSE 315

Peripherals & Interfacing

Abdullah Al Omar

Lecturer, CSE, UAP

What is Arduino

- Arduino board is an open-source platform used to make electronics projects.
- It consists of both a microcontroller and a part of the software or Integrated Development Environment (IDE) that runs on your PC.
- It is used to write & upload computer code to the physical board.
- The platform of an Arduino has become very famous with designers or students just starting out with electronics, and for an excellent cause.

Different types of Arduino Boards

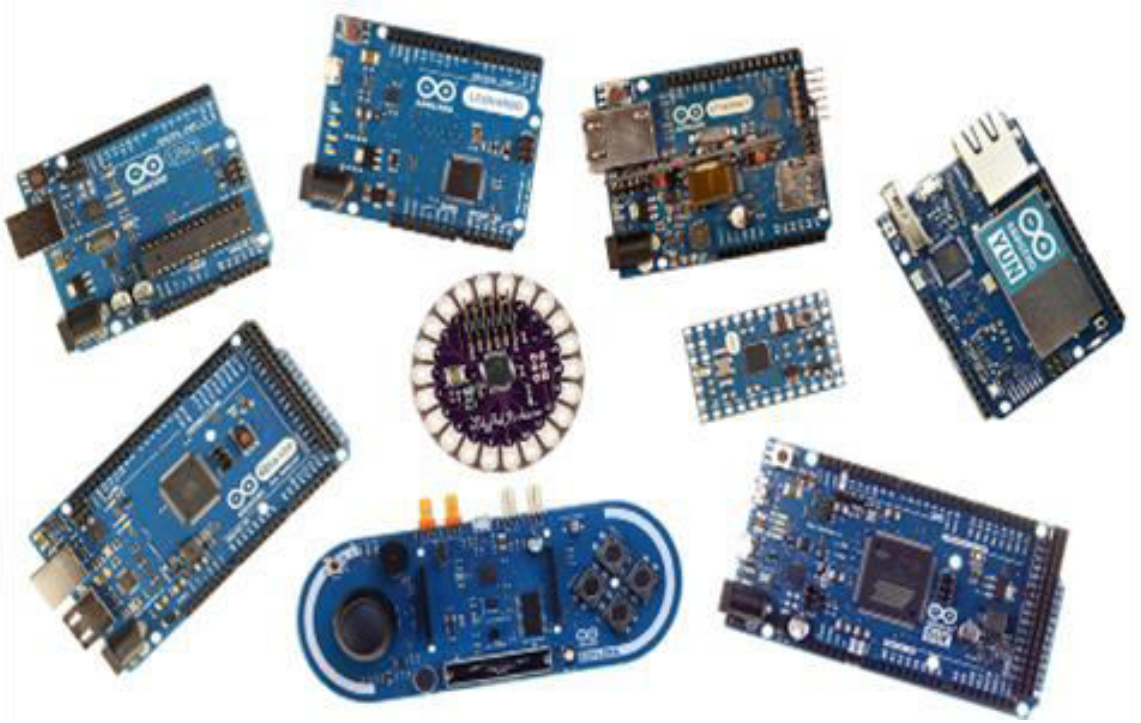


Fig 1. Different types of Arduino Boards.

Basic Features:

- The Arduino does not require a separate part of hardware.
- In order to program a new code onto the board you can just use a USB cable.
- The Arduino IDE uses a basic version of C++, making it simpler to learn the program.
- At last, Arduino board offers a typical form factor that breaks out the functions of the microcontroller into a more available package.

Why Arduino:

- Arduino board has been used for making different engineering projects and different applications.
- The Arduino software is very simple to use for beginners, yet flexible adequate for advanced users.
- It runs on windows, Linux and Mac.
- Teachers and students in the schools utilize it to design low cost scientific instruments to verify the principles of physics and chemistry.
- Arduino also makes simpler the working process of microcontroller.

In a Nutshell:

- Inexpensive
- Cross-platform
- Simple, clear programming environment
- Open source and extensible software
- Open source and extensible hardware

Feature analogy of different boards:

Arduino Boards	Processor	Memory	Digital I/O	Analogue I/O
Arduino Uno	16Mhz ATmega328	2KB SRAM, 32KB flash	14	6 input, 0 output
Arduino Due	84MHz AT91SAM3X8E	96KB SRAM, 512KB flash	54	12 input, 2 output
Arduino Mega	16MHz ATmega2560	8KB SRAM, 256KB flash	54	16 input, 0 output
Arduino Leonardo	16MHz ATmega32u4	2.5KB SRAM, 32KB flash	20	12 input, 0 output

List of Arduino boards

- Arduino Uno (R3)
- LilyPad Arduino
- Red Board
- Arduino Mega (R3)
- Arduino Leonardo

Arduino Uno R3

- The Uno is a huge option for your initial Arduino.
- It consists of 14-digital I/O pins, where 6-pins can be used as PWM(pulse width modulation outputs).
- 6-analog inputs, a reset button, a power jack, a USB connection and more.
- It includes everything required to hold up the microcontroller.
- Simply attach it to a PC with the help of a USB cable and give the supply to get started with a AC-to-DC adapter or battery.

Arduino Uno(R3)

Fig 2. Arduino Uno basic hardware

Some projects based on Arduino UNO:

- Automatic Medicine Reminder Using Arduino
- Obstacle Avoiding Robot using Arduino and Ultrasonic Sensor
- Google Assistant Based Voice Controlled Home Automation using DIY Arduino Wi-Fi Shield
- Self balancing robot
- Line follower

Arduino Lilypad

- The Lily Pad Arduino board is a wearable e-textile technology .
- It expanded by Leah Buechley and considerately designed by “Leah and SparkFun”.
- Each board was imaginatively designed with huge connecting pads
- A smooth back to let them to be sewn into clothing using conductive thread.
- This Arduino also comprises of I/O, power, and also sensor boards which are built especially for e-textiles. These are even washable

Arduino Lilypad

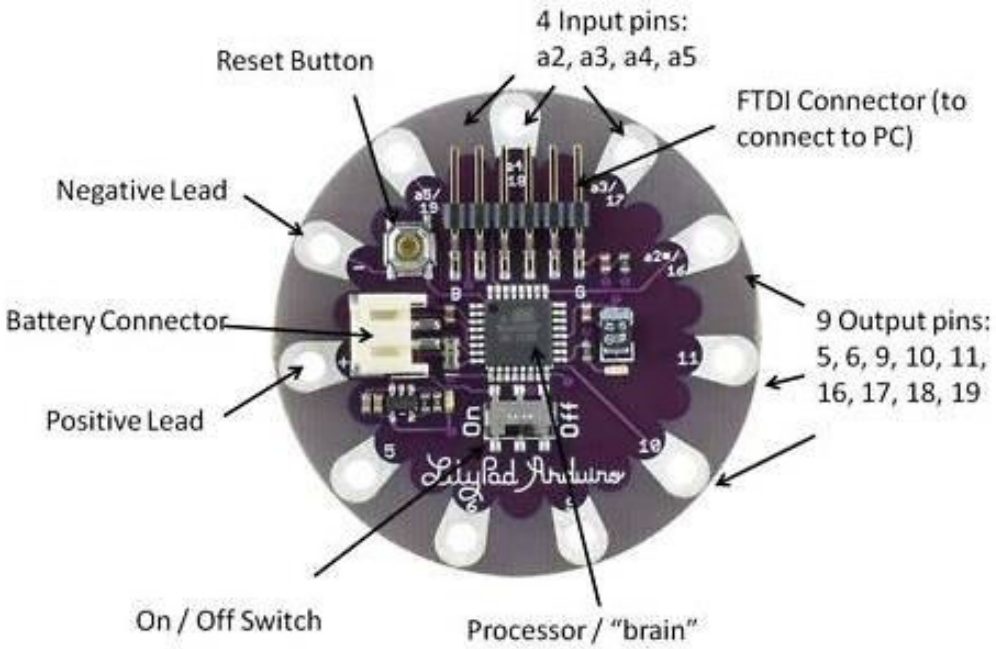


Fig 3. Arduino Lilypad basic hardware

Some projects based on Arduino Lilypad:

Some projects based on Arduino Lilypad:

- Unicorn Horn With NeoPixel LEDs and Arduino Lilypad
- Arduino LilyPad Controlled NeoPixel Earrings
- Hertzian Armor

Arduino Redboard

- The RedBoard Arduino board can be programmed using a Mini-B USB cable using the Arduino IDE.
- It will work on Windows 8 without having to modify your security settings.
- It is more constant due to the USB or FTDI (Future Technology Device International) chip we used and also it is entirely flat on the back.
- Creating it is very simple to utilize in the project design.
- Just plug the board, select the menu option to choose an Arduino Redboard and you are ready to upload the program.
- You can control the RedBoard over USB cable using the barrel jack.

Arduino Redboard

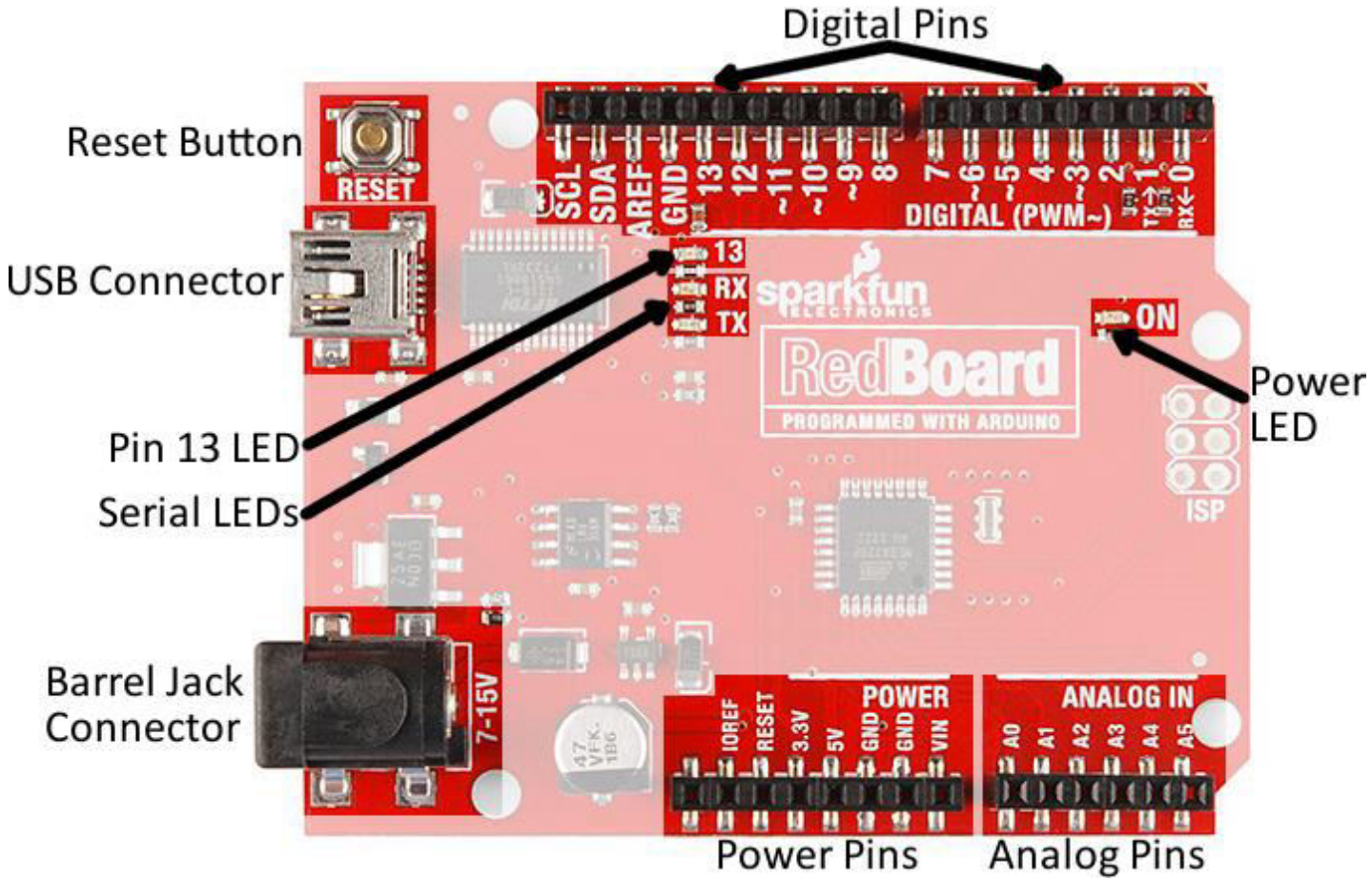


Fig 4. Arduino Redboard basic hardware

Some projects based on Arduino Redboard:

- DC MOTORS AND MOTOR DRIVERS
- Autonomous Driving Vehicle

Arduino Mega (R3)

- The Arduino Mega is similar to the UNO's big brother.
- It includes lots of digital I/O pins (from that, 14-pins can be used as PWM o/ps).
- It includes 6-analog inputs, a reset button, a power jack, a USB connection and a reset button.
- It includes everything required to hold up the microcontroller.

Arduino Mega (R3)

- Simply attach it to a PC with the help of a USB cable.
- Give the supply to get started with a AC-to-DC adapter or battery.
- The huge number of pins make this Arduino board very helpful for designing the projects .
- It need a bunch of digital i/ps or o/ps like lots buttons.

Arduino Mega (R3)

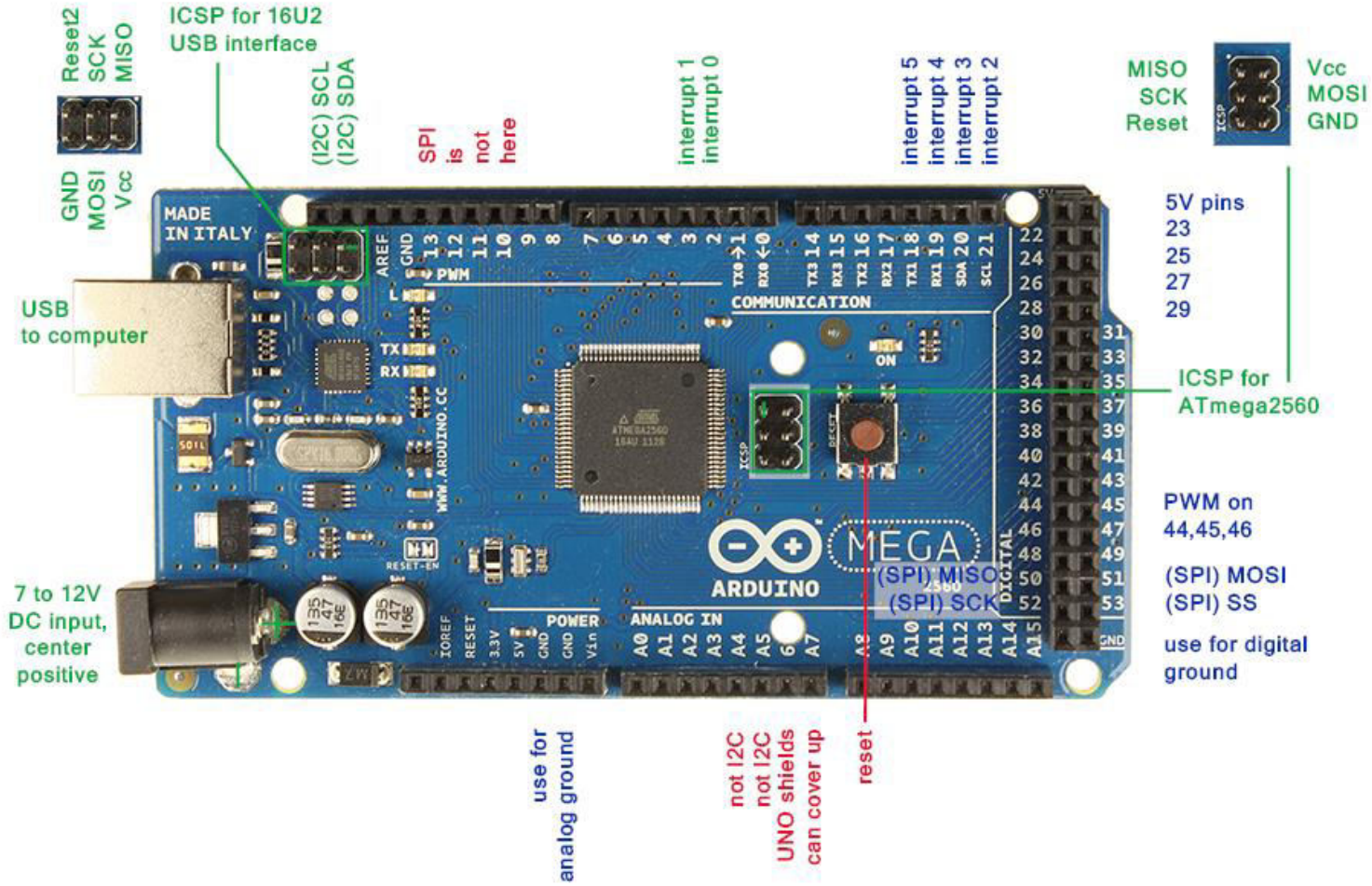


Fig 5. Arduino Mega (R3) basic hardware

Some projects based on Arduino Mega:

- Arduino Serial Communication
- Arduino Solar Tracker
- Arduino RFID Reader
- Frequency Counter Using Arduino

Arduino Leonardo:

- The first developed board of an Arduino series is the Leonardo board.
- This board uses one microcontroller along with the USB.
- It can be very simple and cheap also.
- This board handles USB directly.
- The program libraries are obtainable which let the Arduino board to follow a keyboard of the computer, mouse, etc.

Arduino Leonardo:

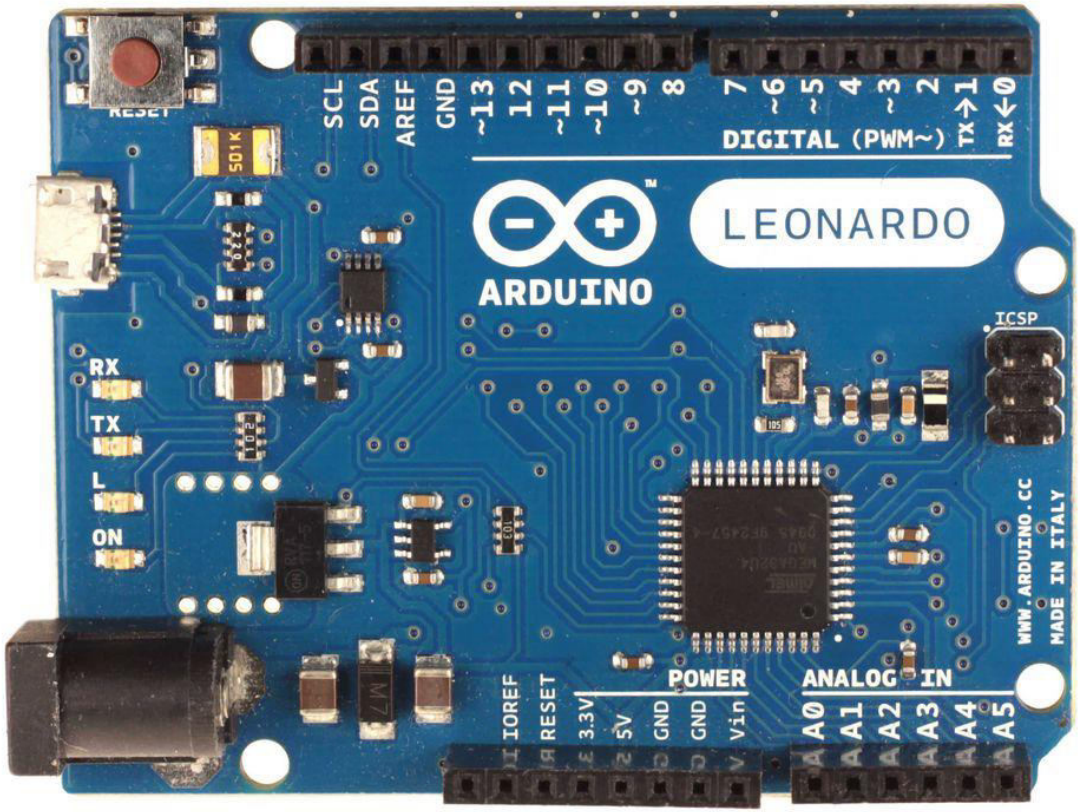


Fig 6. Arduino Leonardo basic hardware

Some projects based on Arduino leonardo:

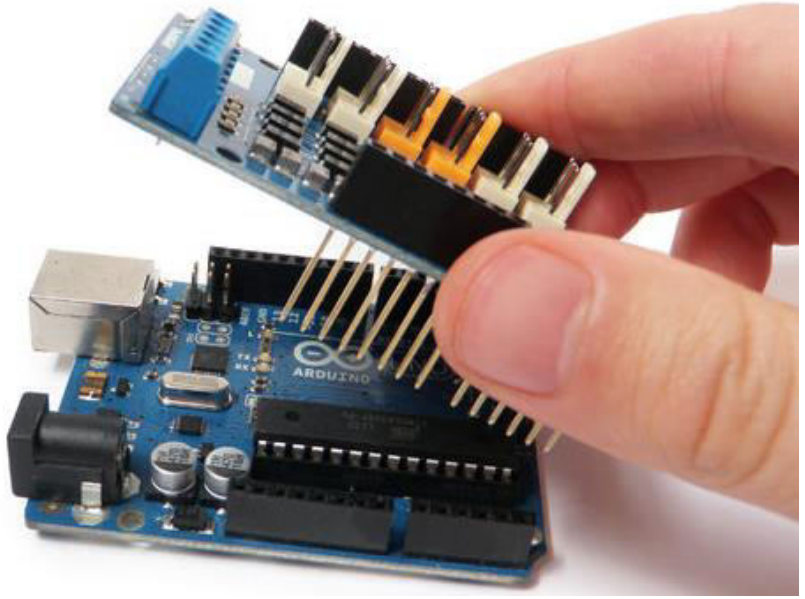
- Humanoid Arm
- Arduino PC Monitor
- Autonomous Navigation and 2D Mapping
- Arduino PowerPoint Pointer
- Rangefinder for Garage Parking with Arduino

Arduino Shields:

- Arduino shields are pre built circuit boards used to connect to a number of Arduino boards.
- These shields fit on the top of the Arduino compatible boards
- It provide an additional capabilities like connecting to the following:
  - Internet
  - Motor controlling
  - Providing wireless communication
  - LCD screen controlling

Arduino Shields:

- Shields are useful to extend the capacity of your Arduino.



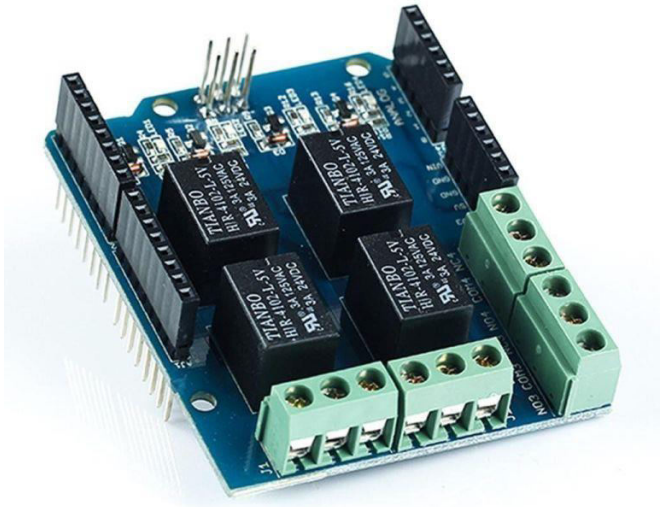
Different types of Shields:

- **1. Ethernet Shield**

The Ethernet Shield allows you to connect your Arduino to the internet. You just have to plug the shield onto the Arduino board and then, connect it to your network.

- **2. Relay Shield**

The Relay Shield is a module with 4 mechanical relays that provides you an easy way to control high voltage.

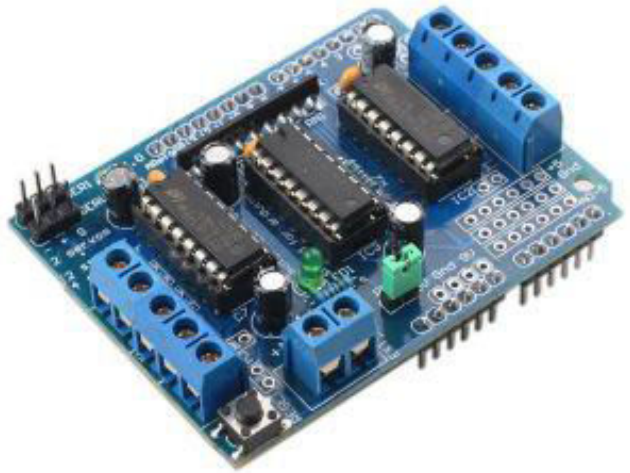


- **3. ProtoShield**

The ProtoShield is a prototyping Shield that makes it easy to prototype. It allows for easy connections between the breadboard and the Arduino.

- **4. Motor Shield**

This Shield allows an easy control of motor direction and speed. It makes it easy to incorporate a motor into any of your projects.

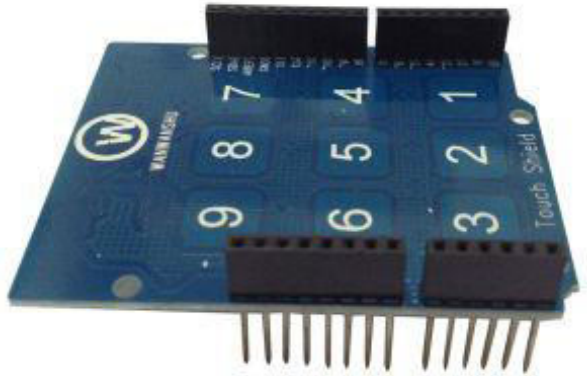


- **5. LCD Shield**

This Shield makes it easy to use a 16×2 Character LCD. With this, it is possible to control a 16×2 Character LCD, up to 3 backlight pins and 5 keypad pins using only the two I2C pins on the Arduino.

- **6. Capacitive Touchpad Shield**

The Touchpad Shield allows you to build simple capacitive touch interfaces.



- **7. Smoke Detector Shield**

This Shield can detect concentrations of combustible gas in the air and read it as an analogue value. Useful to make a smoke detector system.

- **8. 64-Button Shield**

With this Shield you can connect up to 64 buttons to your Arduino. Some cool projects with this shield include musical instruments, cool computer interfaces, etc.

- **9. Joystick Shield Kit**

The joystick Shield provides simple analog inputs and four separate buttons and one button under the joystick itself.

- **10. GSM/GPRS Shield**

The GSM/GPRS Shield allows you to connect your Arduino to GSM/GPRS cell phone network. It allows you to dial a phone number or send a text message to a friend via easy to use AT commands.

- 11. **GPS Logger Shield**
- 12. **Wireless SD Shield**
- 13. **cc300 Wi-Fi Shield**
- 14. **HC-05 Bluetooth Shield**
- 15. **MP3 Player Shield**

You must know at least twenty-twenty five Arduino shields with it functionalities.

Thank you

Scope & Operators in Arduino

CSE- 315

Peripherals & Interfacing

Abdullah Al Omar

Lecturer, CSE, UAP

Scope

There are three types of scope in Arduino environment.

- Global Scope
- Formal Scope
- Local Scope

Global Scope:

```
Int T , S ;
float c = 0 ; Global variable declaration
void setup () {
}
void loop(){
}
```

Formal Scope:

Suppose void A() is a function:

```
void setup(){
  Serial.begin(9600);
}
void loop() {
                    int i = 2;
                   int j = 3;
                   int k;
                   k = A(i, j); // k now contains 6
                   Serial.println(k);
                   delay(500);
}

int A( int x, int y){
                    int result;
                   result = x * y;
                   return result;
}
```

Formal scope variable:

Local Scope:

- The variable which resides in the function only. So the variable can only be used in the function.
- This variables are not valid in outside of the function.

```
Void loop () {
   int x , y ;
   int z ; //Local variable declaration
   x = 0;
   y = 0; // initialization
   z = 10;
}
```

Operators

There are five types of operator in Arduino:

1. Arithmetic Operator
2. Comparison Operator
3. Boolean Operator
4. Bitwise Operator
5. Compound Operator

Hand Notes

Please make sure to have a pen and paper with you.

- Resource Link- https://www.arduino.cc/reference/en/

Thank you

Control Statements and Loops in Arduino

CSE- 315

Peripherals & Interfacing

Abdullah Al Omar

Lecturer, CSE, UAP

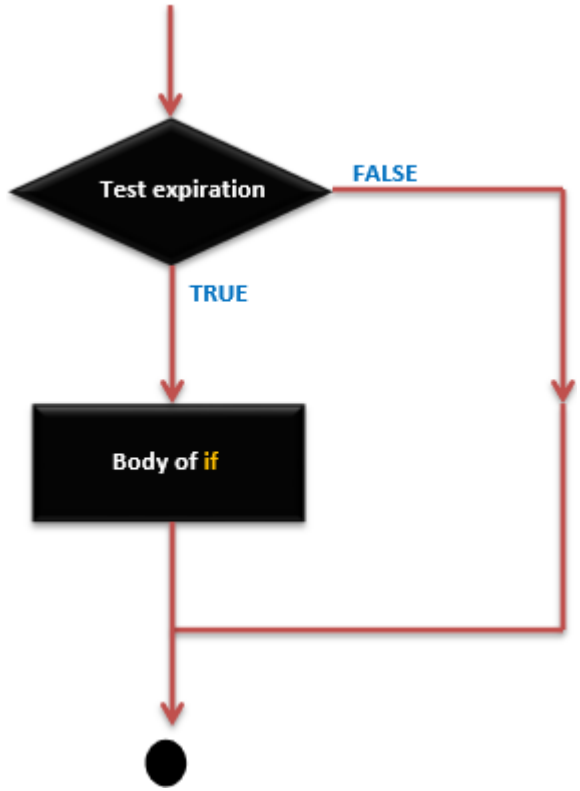
Control statements

Control statement:

- If
- If…. else
- If….else if()….else
- Switch case
- Conditional Operator **?:**

If statement:

```
if (expression)
   statement;

OR,

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

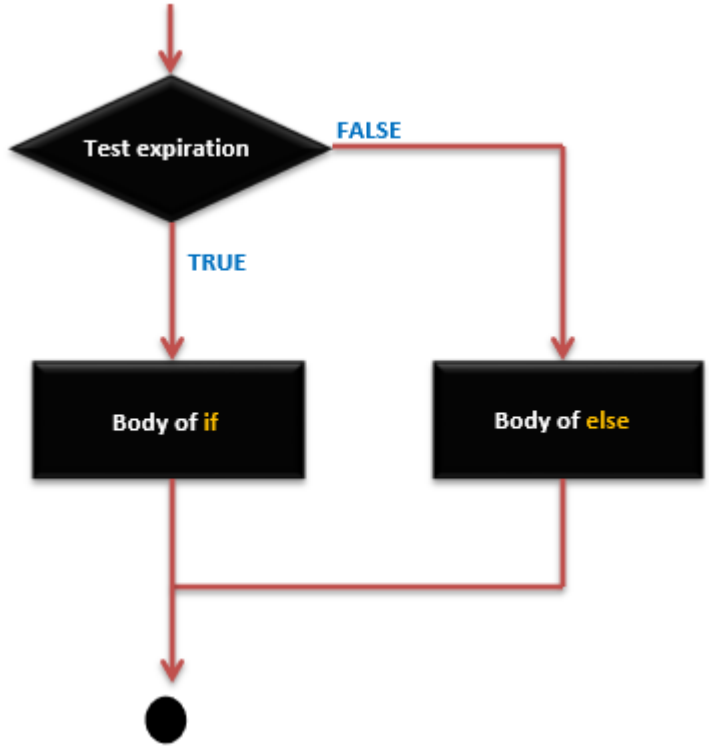


If statement: (contd.)

```
int A = 5 ; /* Global variable definition */
int B = 9 ;
Void setup () {
}
Void loop () {   /* check the boolean condition */
  if (A > B) /* if condition is true then execute the following statement*/
  A++;
  /* check the boolean condition */
  If ( ( A < B ) && ( B != 0 )) /* if condition is true then execute the following statement*/ {
     A += B;
     B--;
  }
}
```

If... else statement:

```
if (expression) {
   Block of statements;
}
else {
   Block of statements;
}
```

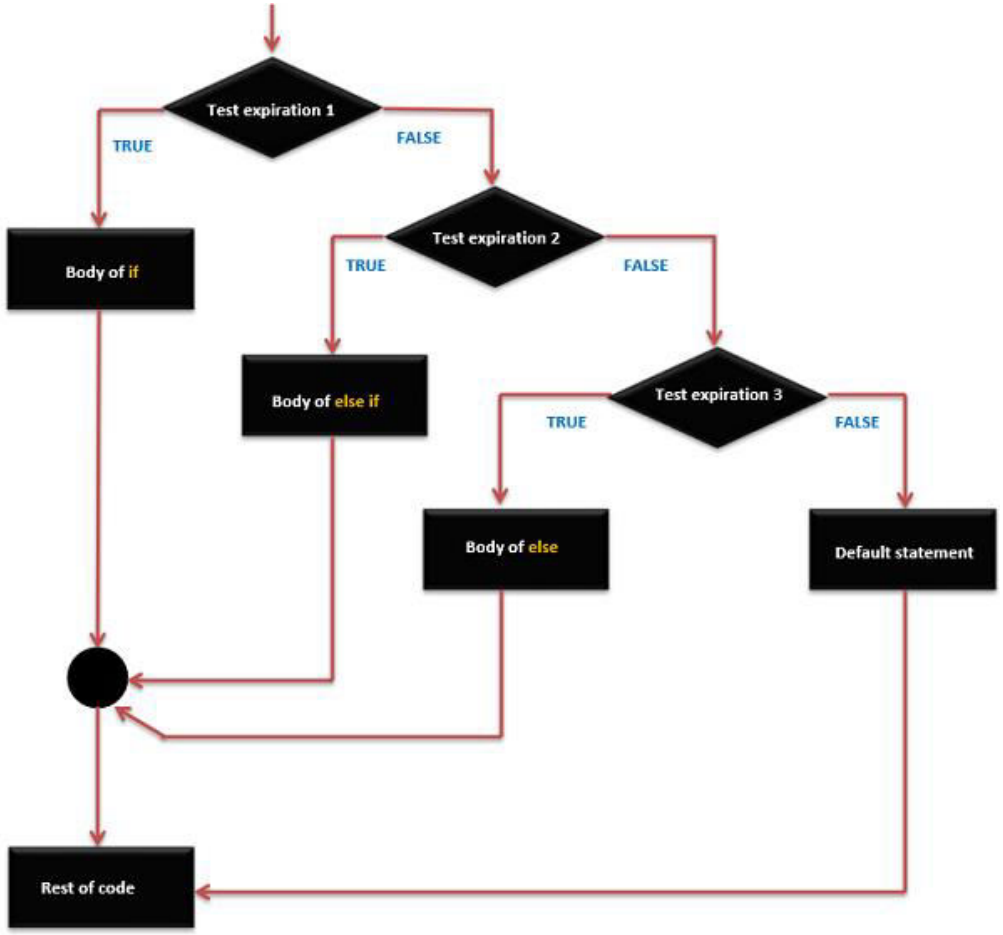


If... else statement: (contd.)

```
int A = 5 ;                /* Global variable definition */
int B = 9 ;
Void setup () {

}
Void loop () {
  /* check the boolean condition */
  if (A > B) /* if condition is true then execute the following statement*/ {
    A++;
  }else {
    B -= A;
  }
}
```

If..else If…. else statement:

```
if (expression_1) {
   Block of statements;
}

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

else {
   Block of statements;
}
```

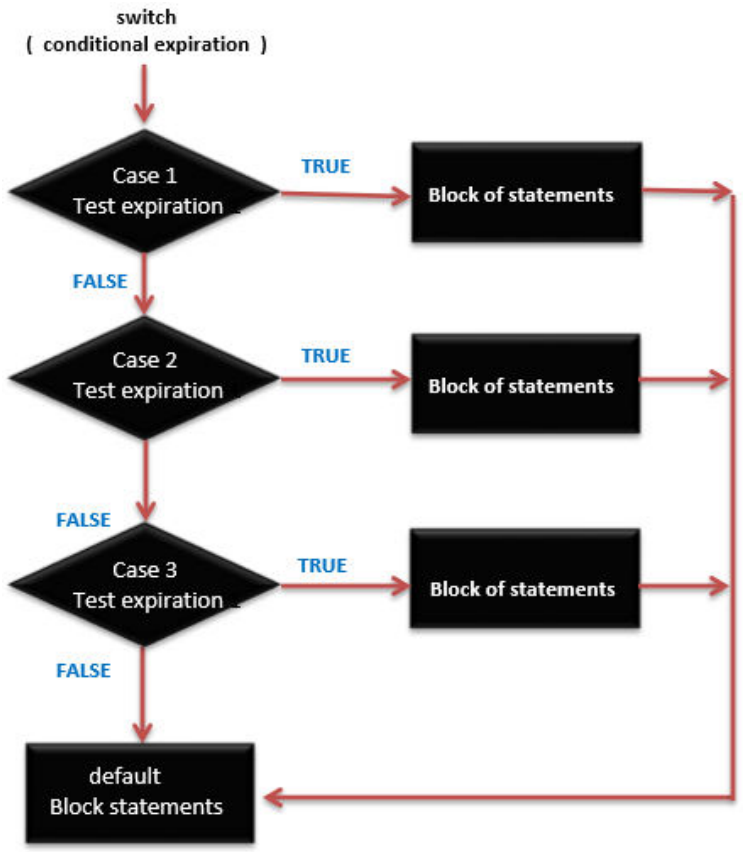


If..else If…. else statement: (contd.)

```
int A = 5 ;       /* Global variable definition */
int B = 9 ;
int c = 15;
Void setup () {
}
Void loop () {
  /* check the boolean condition */
  if (A > B) /* if condition is true then execute the following statement*/ {
    A++;
  }
  /* check the boolean condition */
  else if ((A == B )||( B < c) ) /* if condition is true then
    execute the following statement*/ {
    C = B* A;
  }else
    c++;
}
```

Switch case statement:

```
switch (variable) {
  case label:
  // statements
  break;
}

case label: {
  // statements
  break;
}

default: {
  // statements
  break;
}
```



Switch case statement: (contd.)

```
switch (phase) {
   case 0: Lo(); break;
   case 1: Mid(); break;
   case 2: Hi(); break;
   default: Message("Invalid state!"); break;
}
```

Conditional Operator:

expression1 ? expression2 : expression3

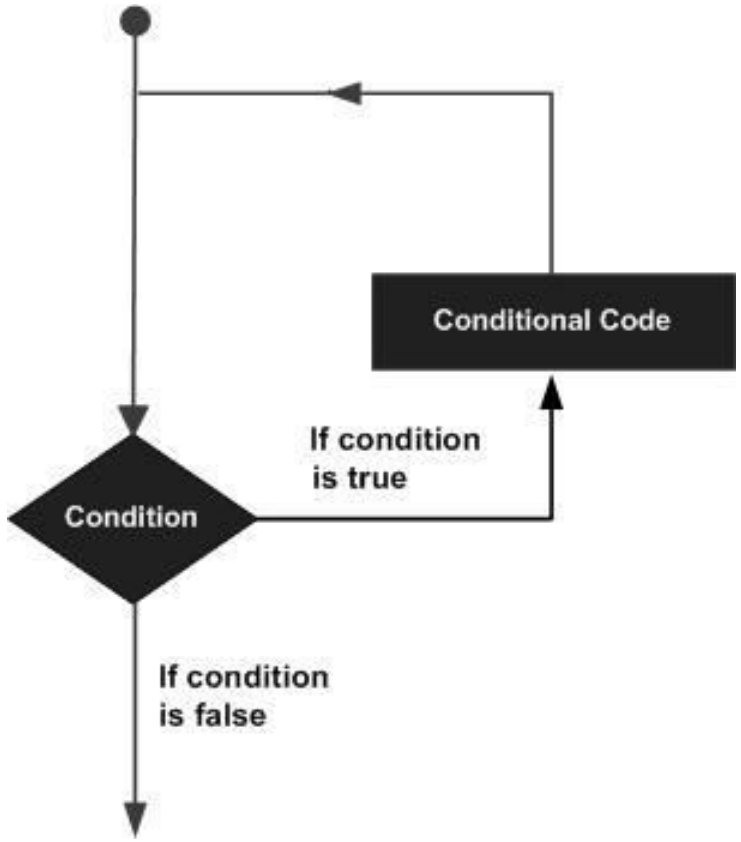
max = (a > b) ? a : b;

If the condition is true then a otherwise b.

Loops

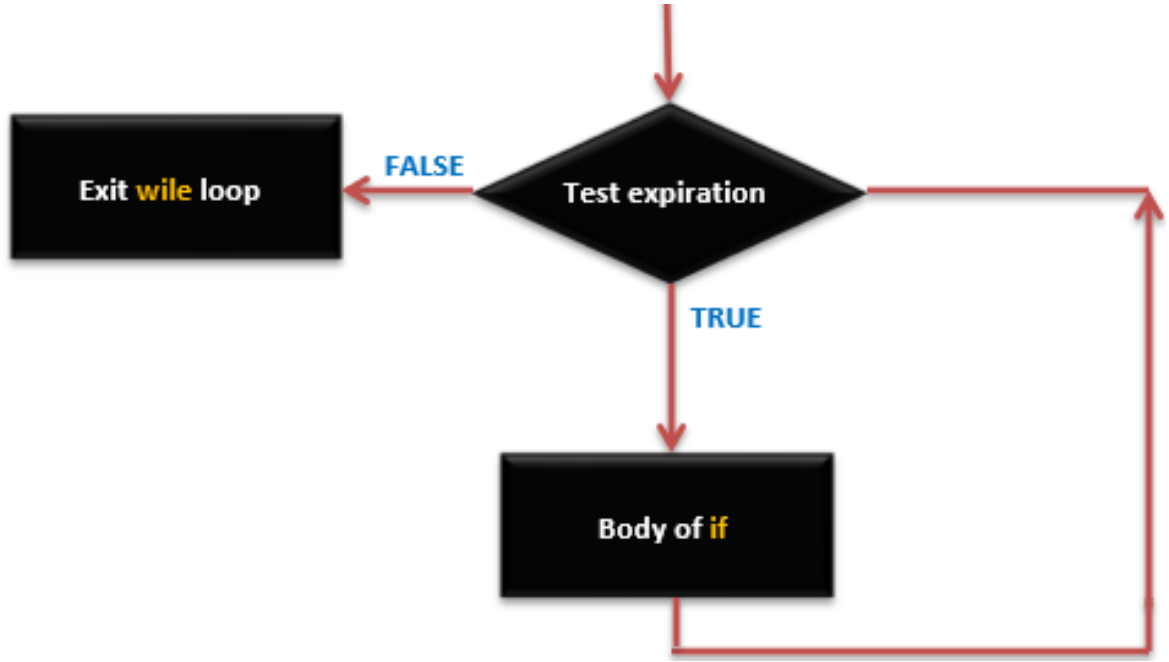
Loops:

- While
- Do…. While
- For
- Nested Loop
- Infinite Loop



While loop:

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

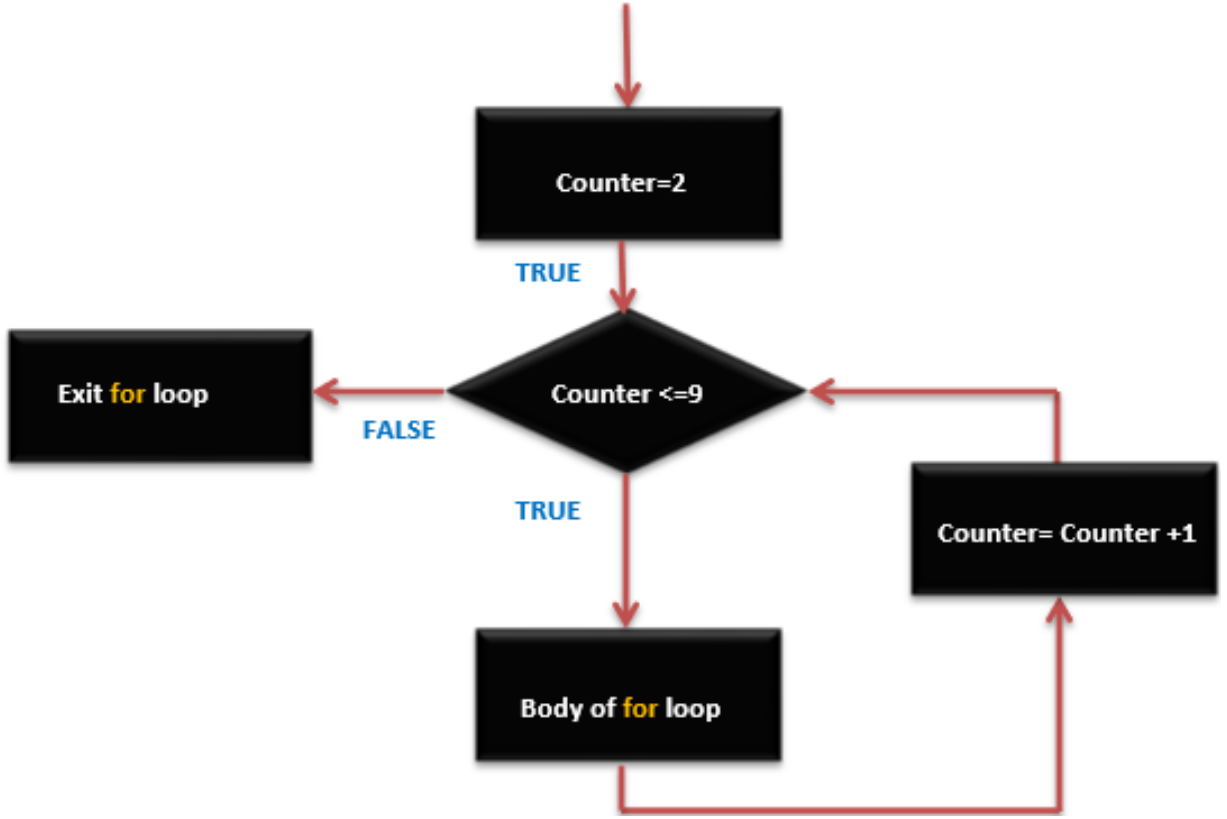


Do… While

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

For loop:

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



Nested Loop:

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

Infinite Loop:

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

while(1) {
   // statement block
}

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

analogWrite()

Syntax-

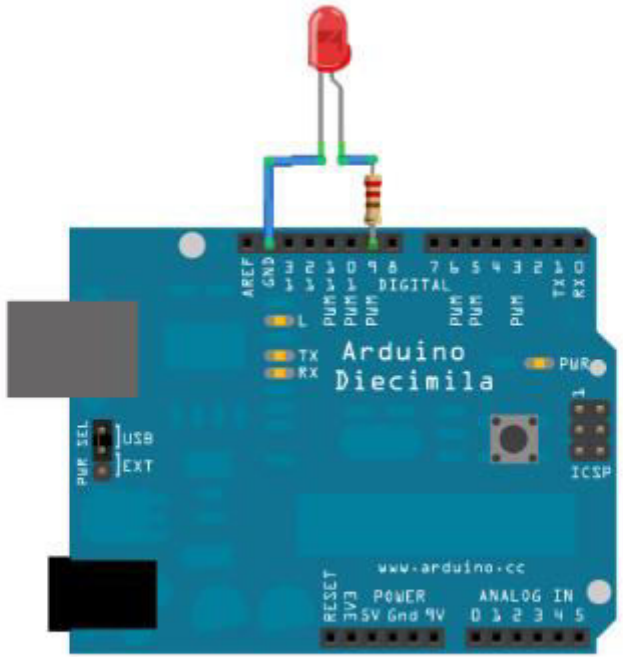
analogWrite(Pin Number, PWM value);

Fading a LED with analogWrite:

Hardware Required:

- Arduino or Genuino board
- LED
- 220 ohm resistor
- hook-up wires
- breadboard

Fading a LED with analogWrite: (Contd.)



Fading a LED with analogWrite: (Contd.)

```
int ledPin = 9;    // LED connected to digital pin 9

void setup() {
  // nothing happens in setup
}
```

Fading a LED with analogWrite: (Contd.)

```
void loop() {
  // fade in from min to max in increments of 5 points:
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
```

Fading a LED with analogWrite: (Contd.)

```
 // fade out from max to min in increments of 5 points:
 for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
   // sets the value (range from 0 to 255):
   analogWrite(ledPin, fadeValue);
   // wait for 30 milliseconds to see the dimming effect
   delay(30);
 }
}
```

Fading a LED with analogWrite: (Contd.) [Loop code at once.]

```
void loop() {
  // fade in from min to max in increments of 5 points:
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += 5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }

  // fade out from max to min in increments of 5 points:
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= 5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```

Thank You

Arduino Uno (R3) Pin Configuration & Installation

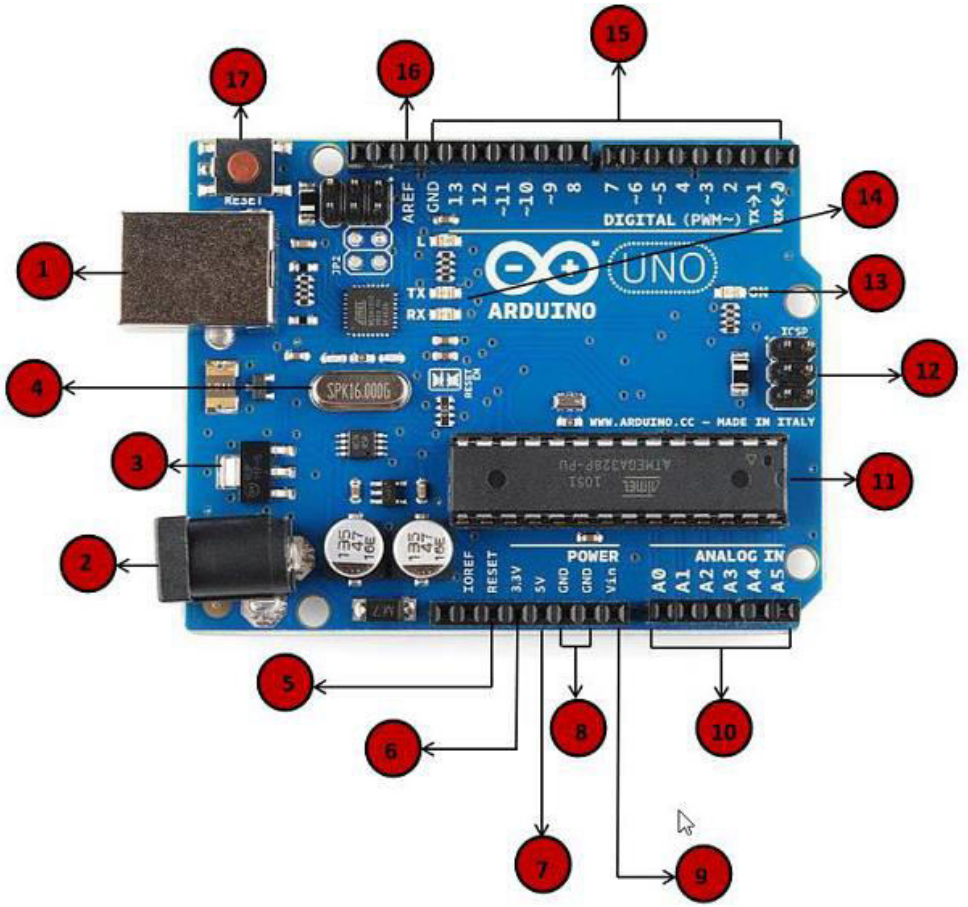
CSE 315

Peripherals & Interfacing

Abdullah Al Omar

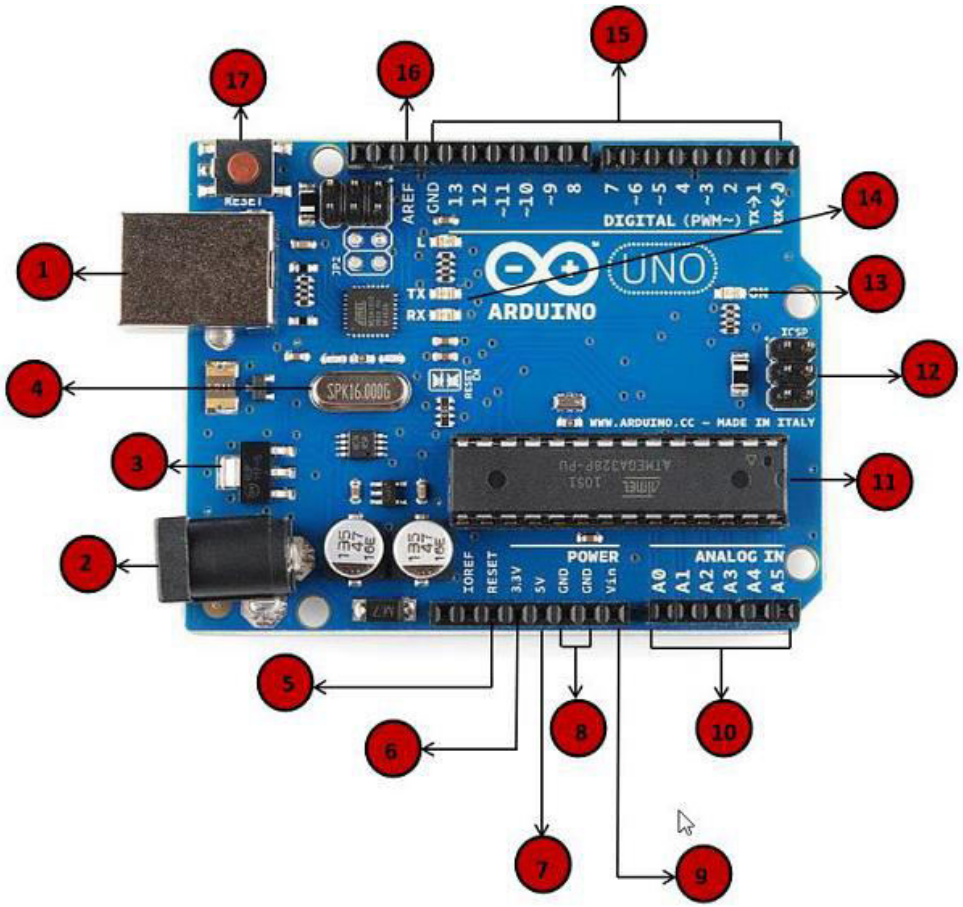
Lecturer, CSE, UAP

Pin Configuration

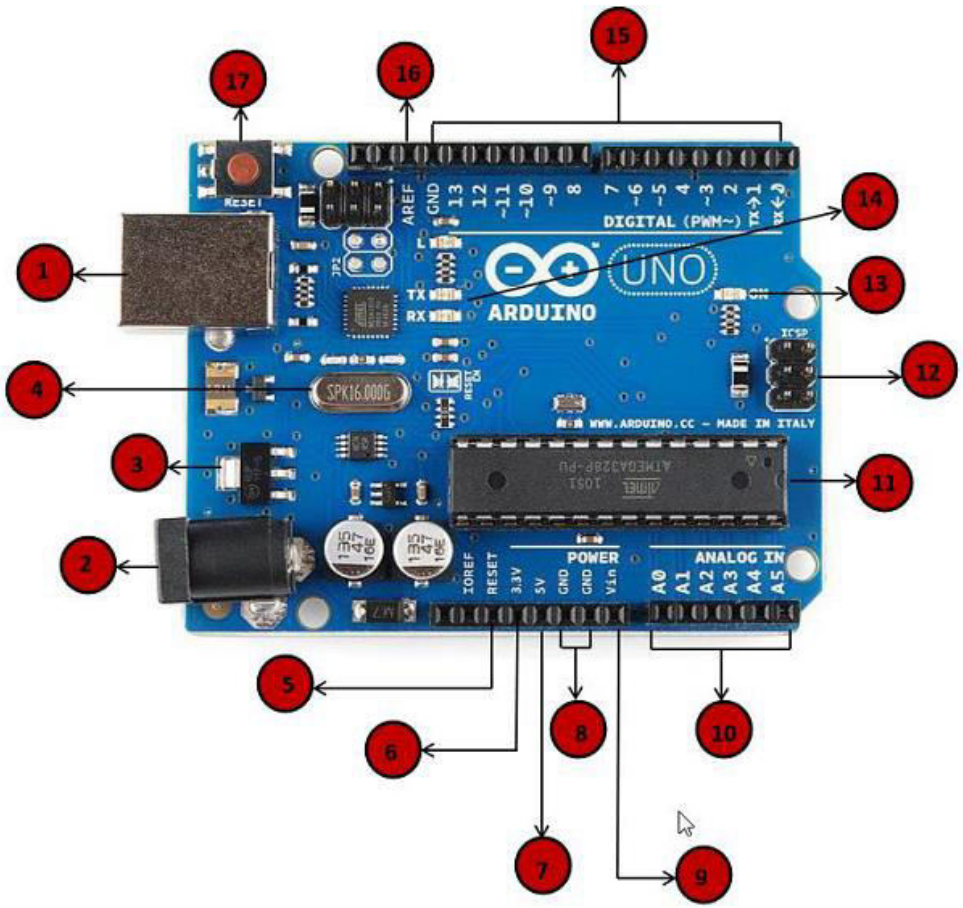


Power USB

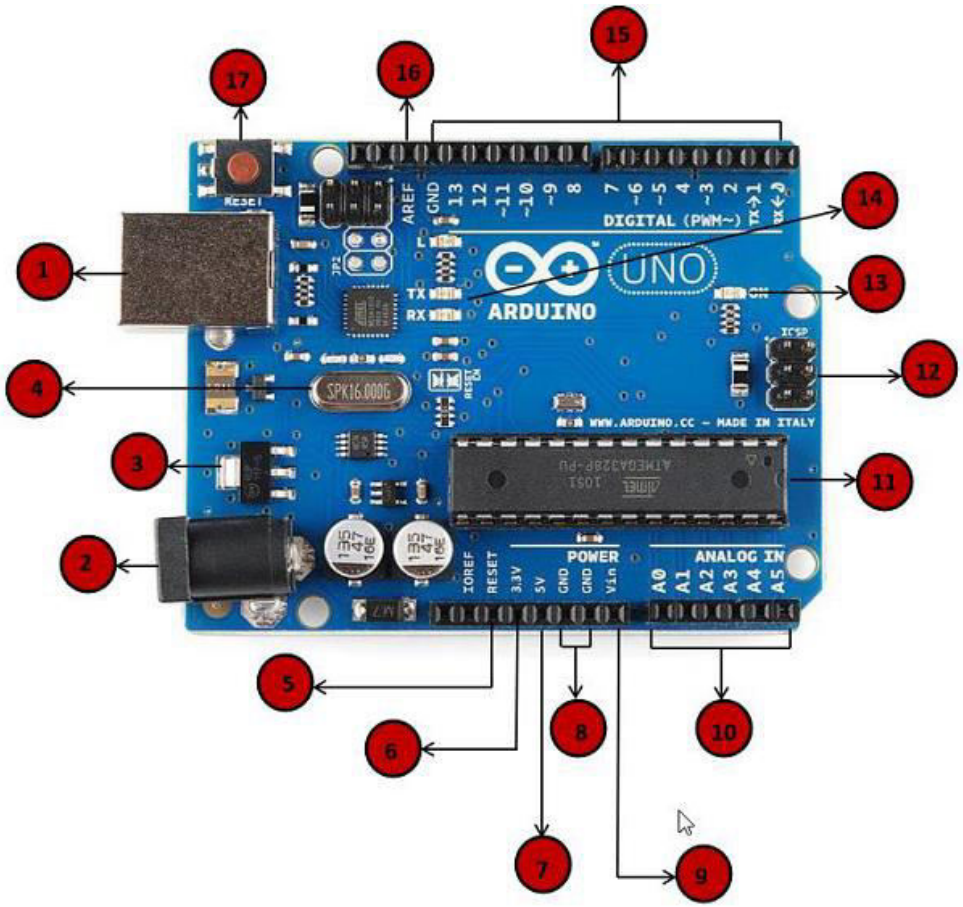
Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).



2	**Power (Barrel Jack)** Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).



3	**Voltage Regulator** The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.



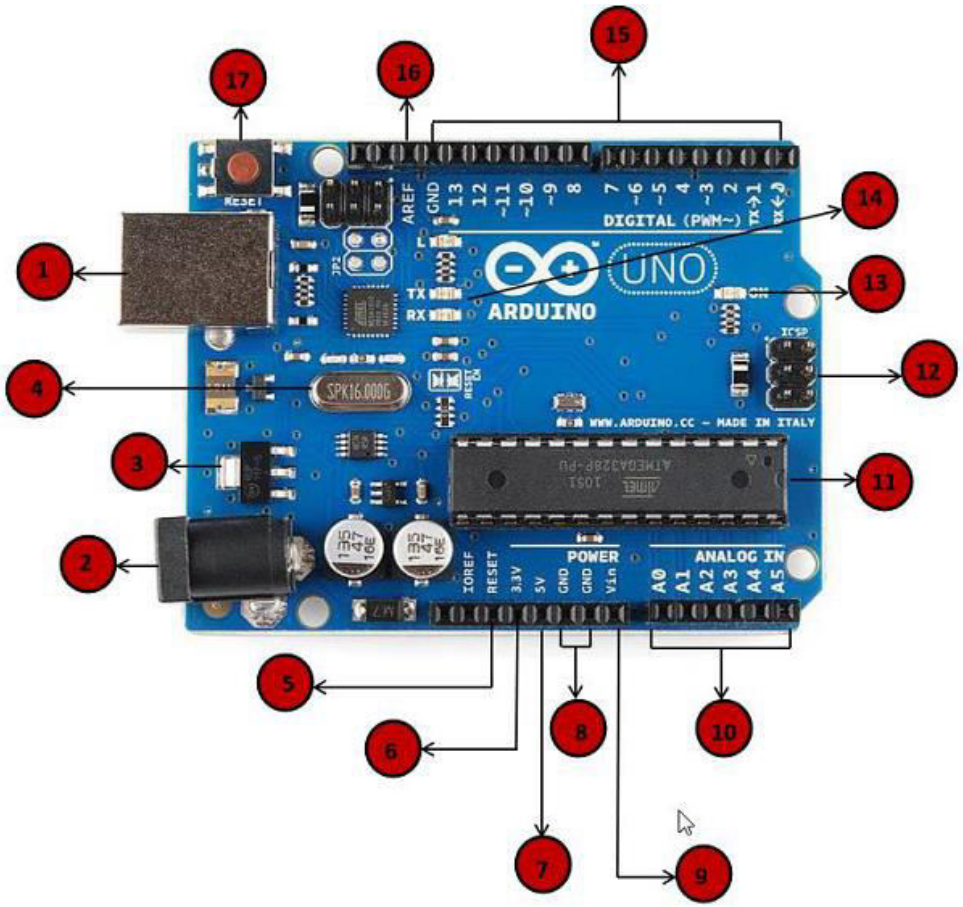
4

Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues.

How does Arduino calculate time?

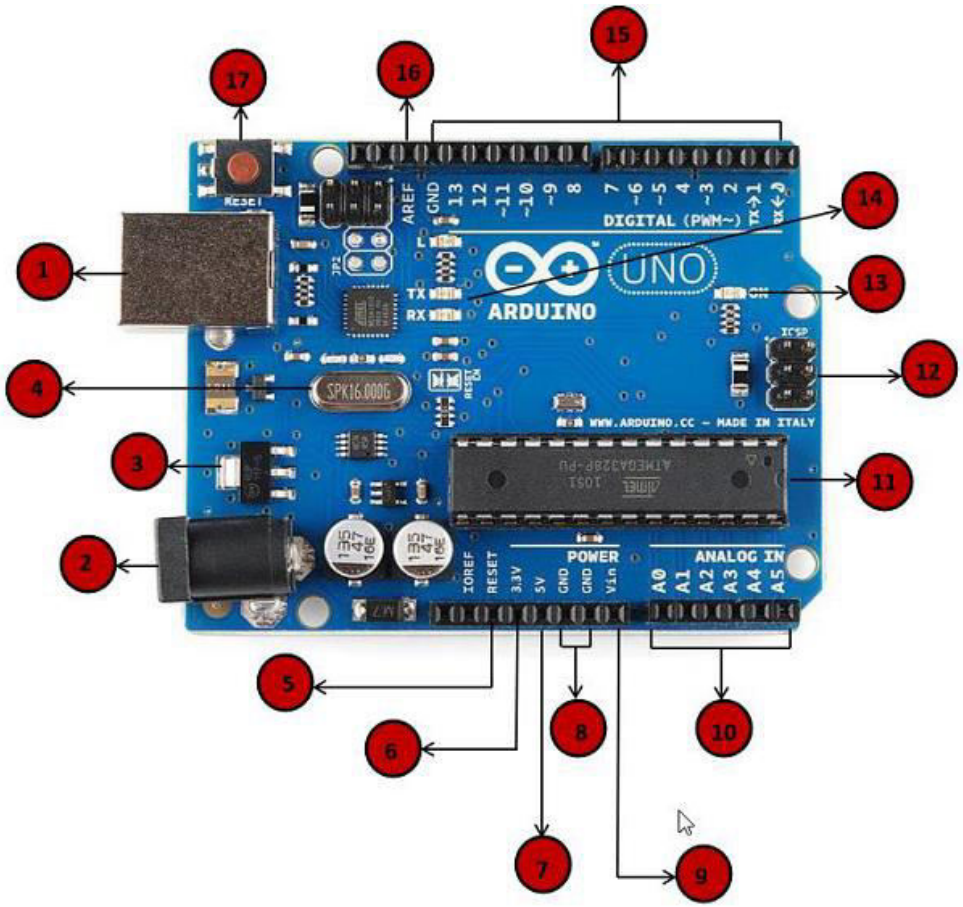
The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.



5,17

Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).



6,7
8,9

Pins (3.3, 5, GND, Vin)

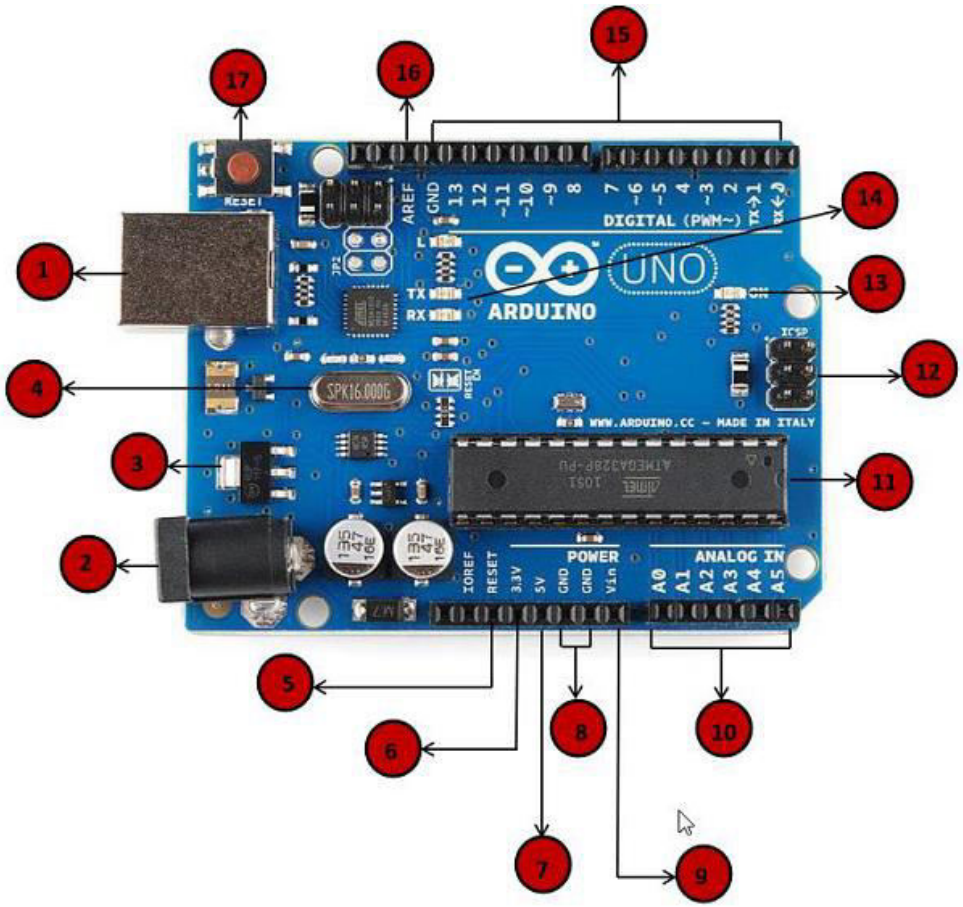
•3.3V (6) – Supply 3.3 output volt

•5V (7) – Supply 5 output volt

•Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.

•GND (8)(Ground) – There are several GND pins on the Arduino, any of which can be used to ground your circuit.

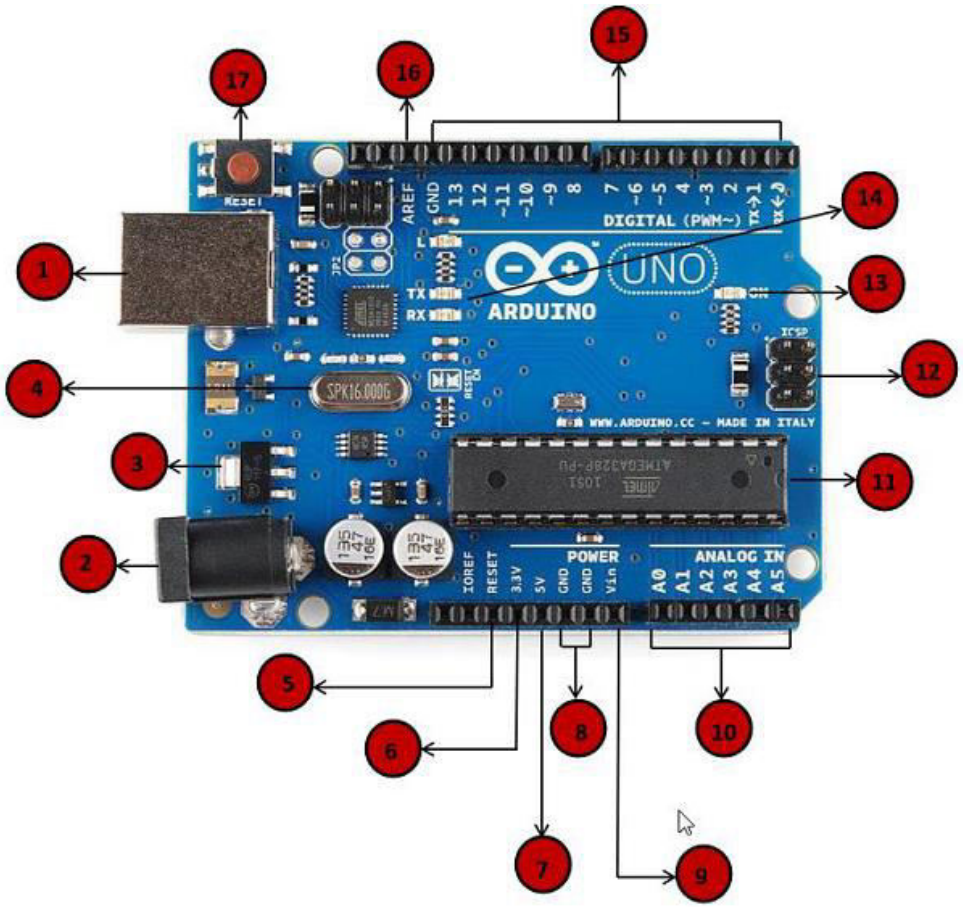
•Vin (9) – This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.



10

Analog pins

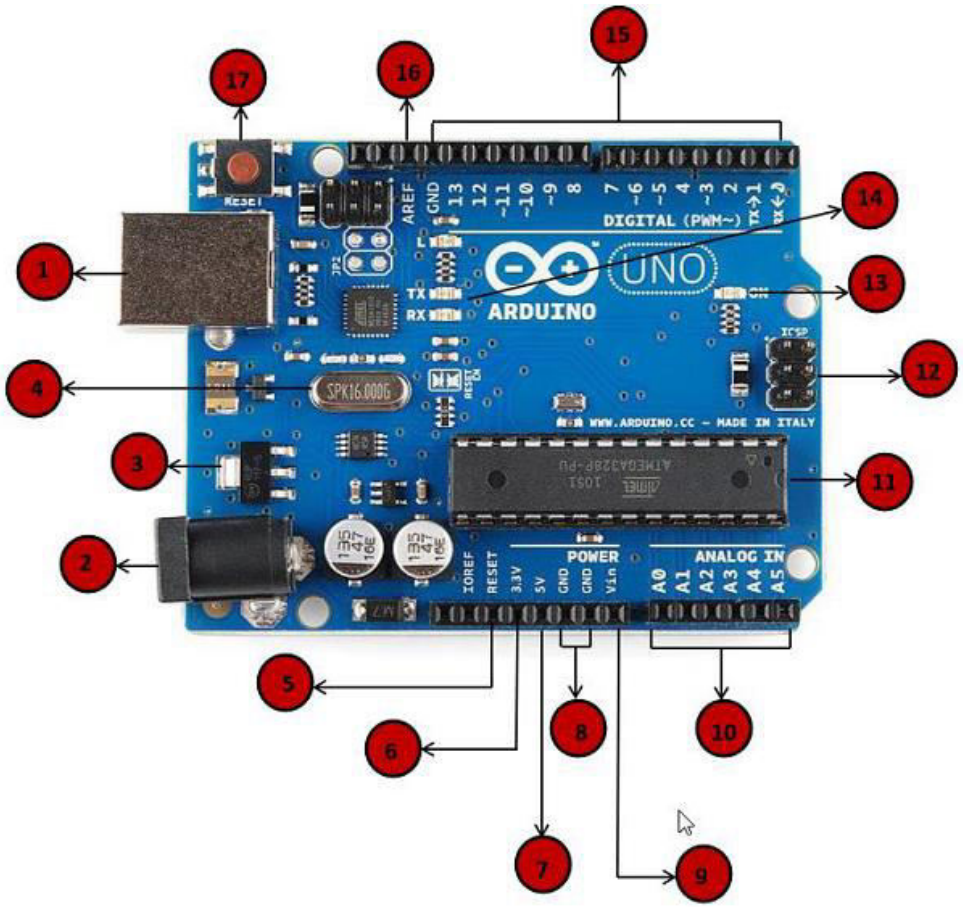
The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.



11

Main microcontroller

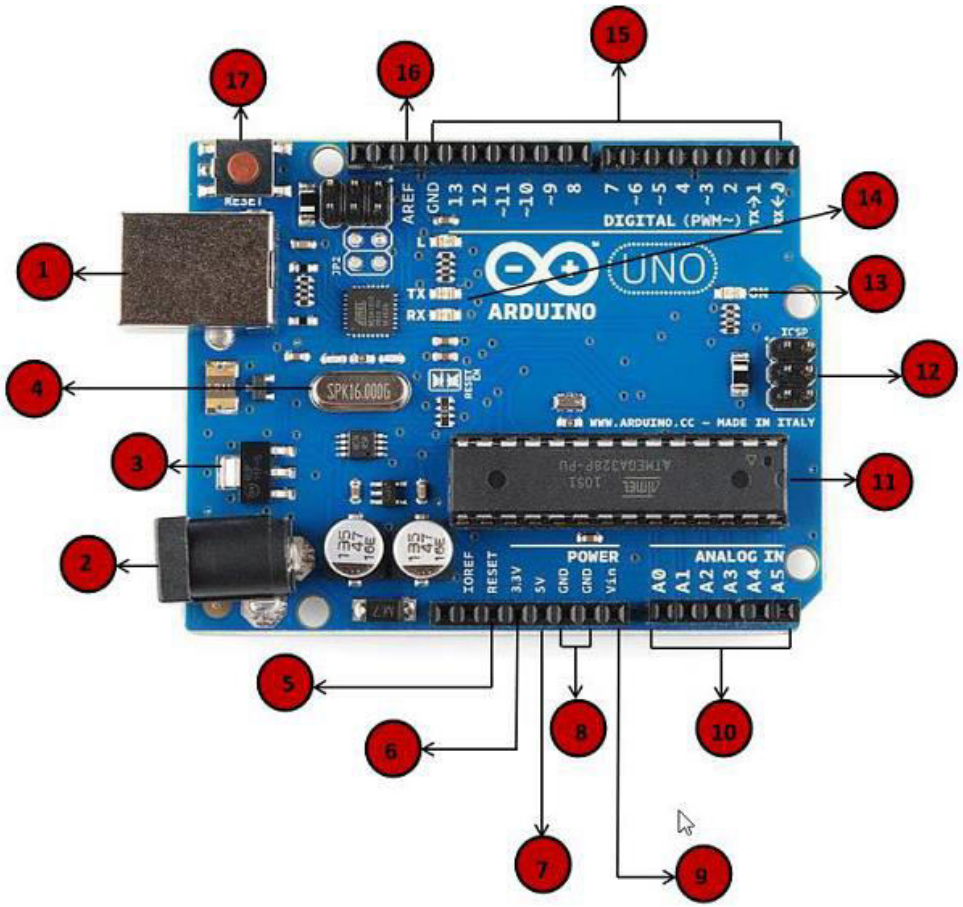
Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.



12

ICSP pin

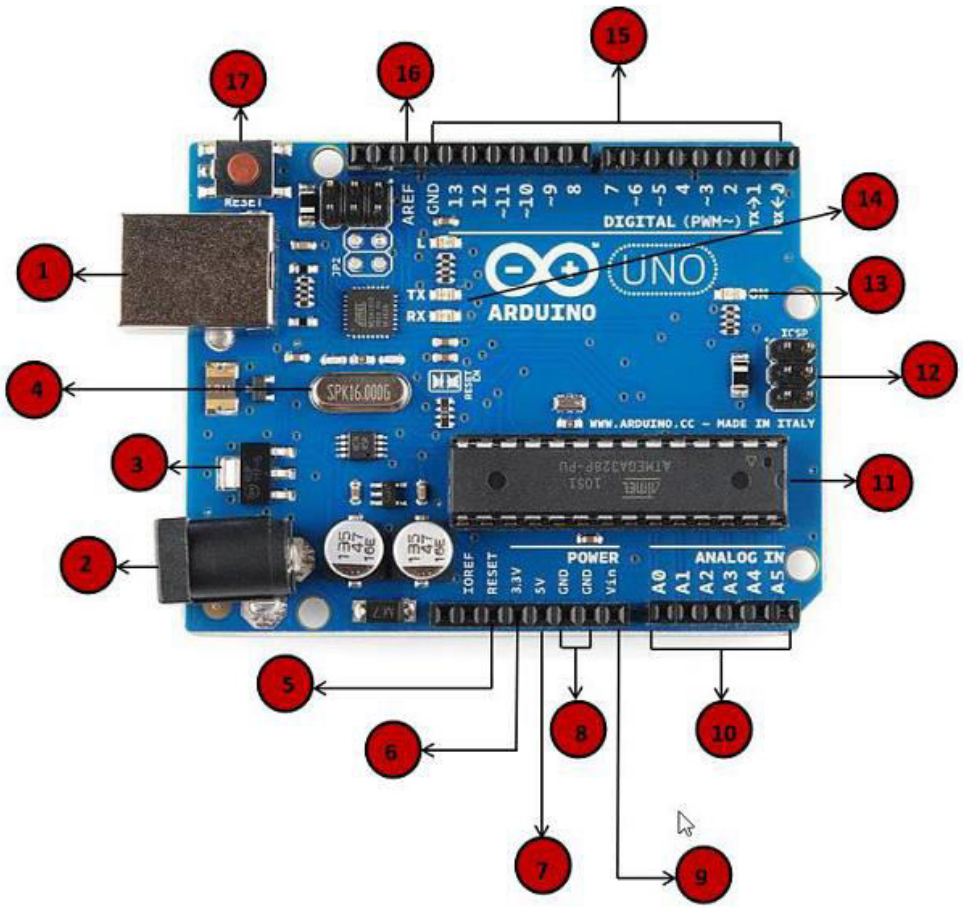
Mostly, ICSP (*In Circuit Serial Programming*) (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.



13

Power LED indicator

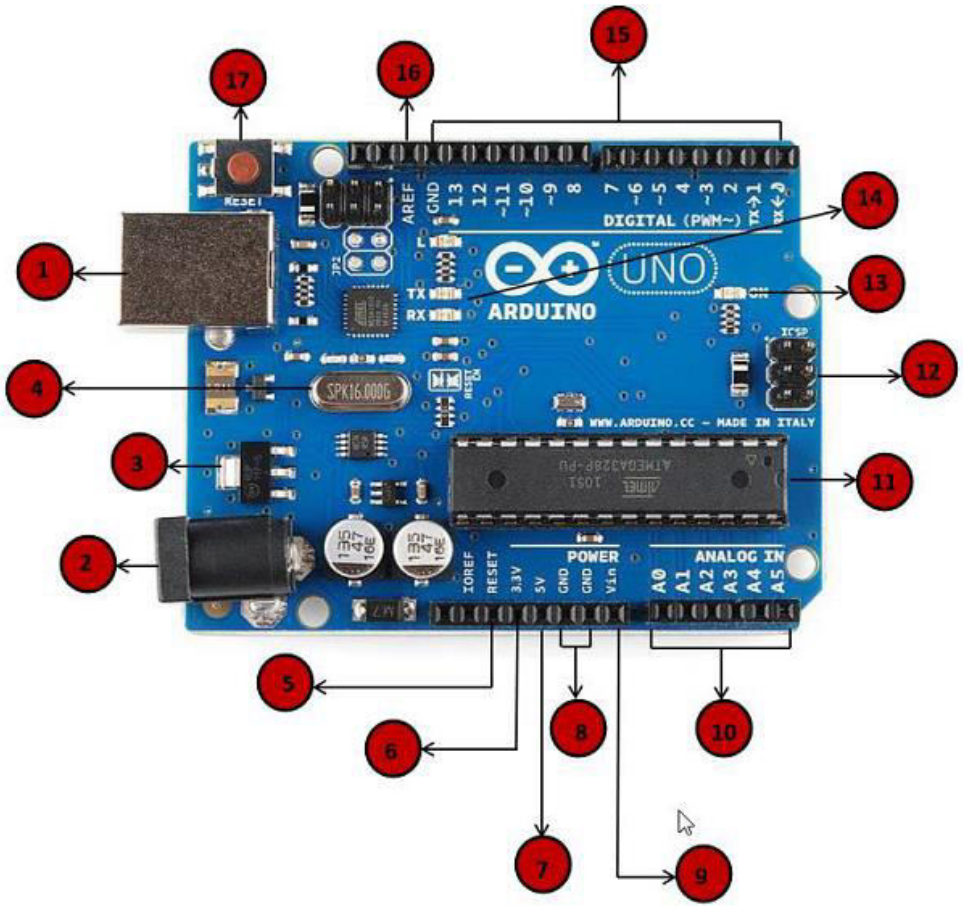
This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.



14

TX and RX LEDs

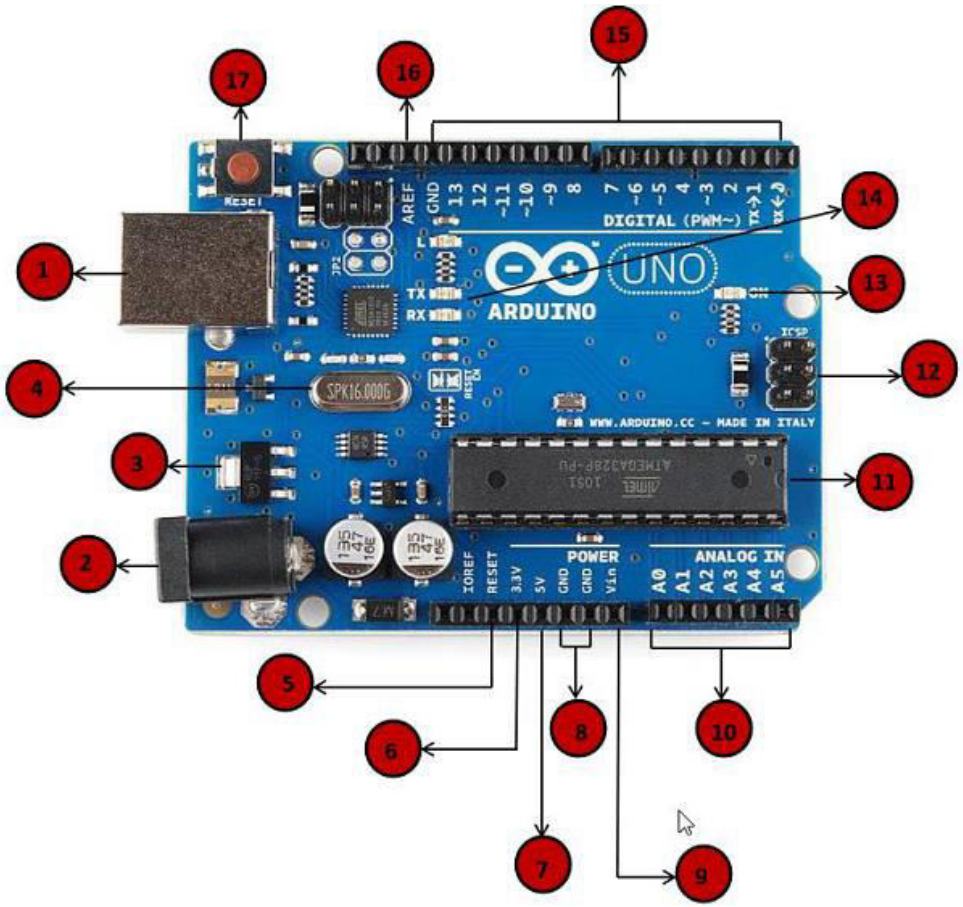
On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.



15

Digital I/O

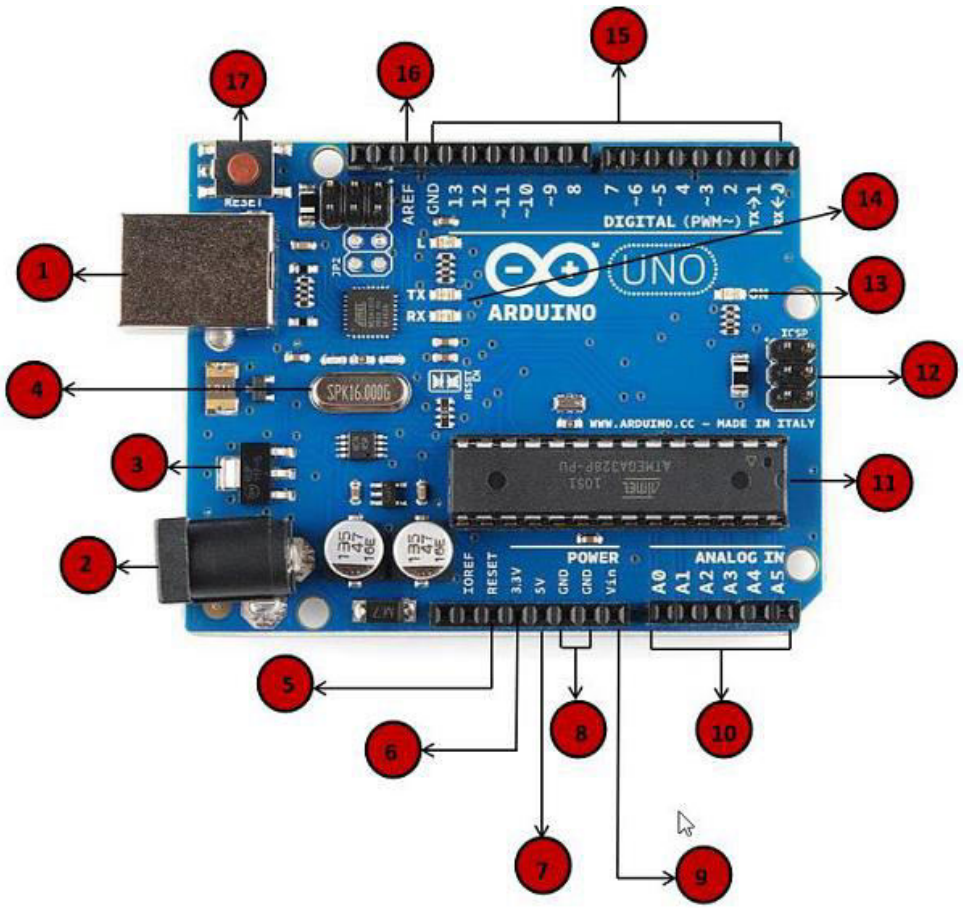
The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.



16

AREF

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.



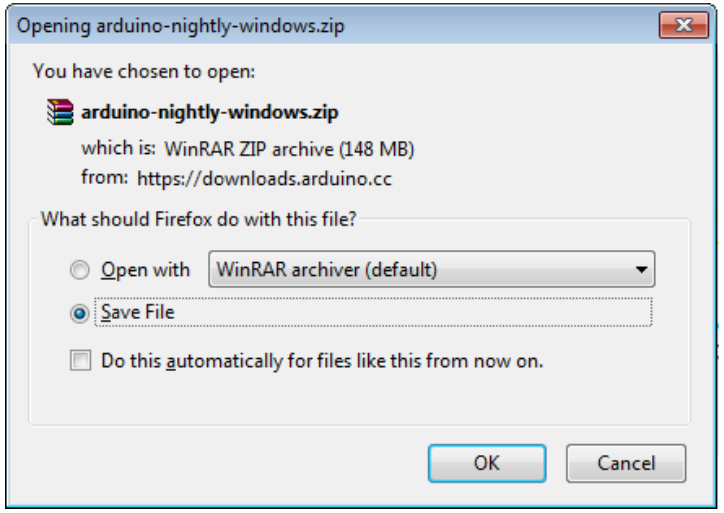
Step 1 – First you must have your Arduino board (you can choose your favorite board) and a USB cable. The kind you would connect to a USB printer as shown in the following image.

USB Cable

- **Step 2** – Download Arduino IDE Software.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

Opening Arduino Nightly Windows

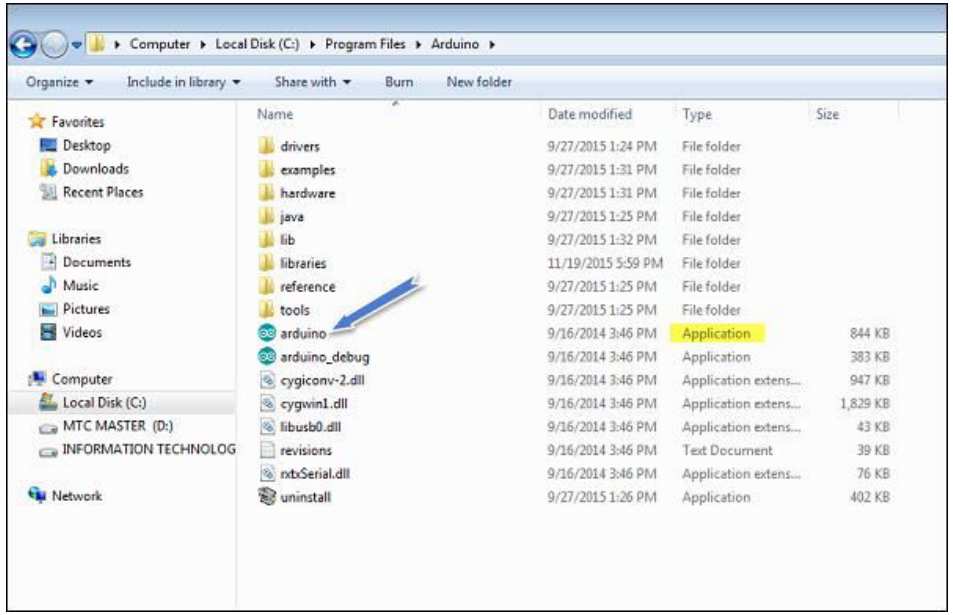


Step 3 – Power up your board.

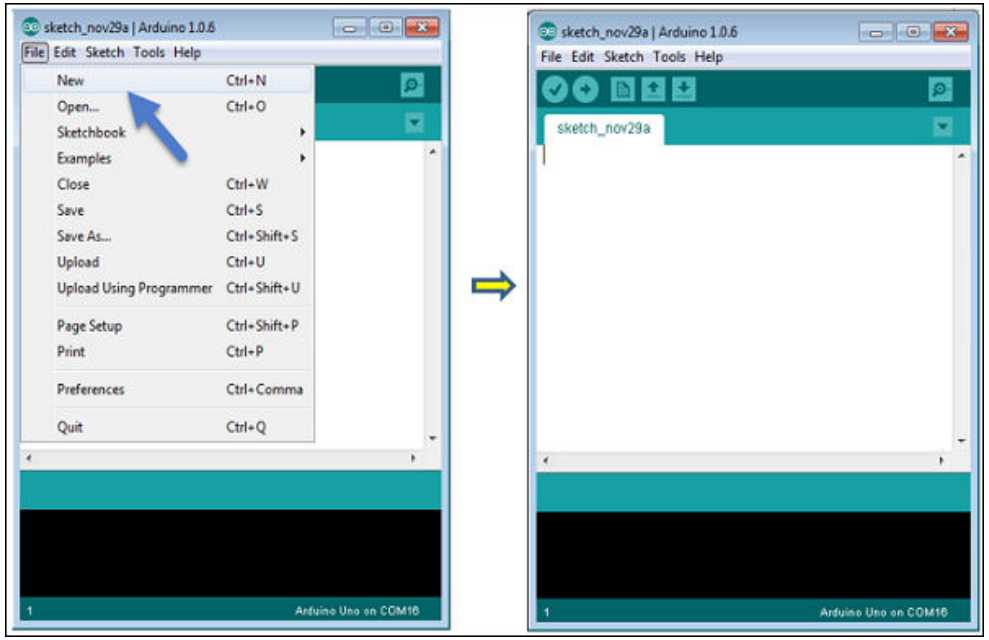
Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 – Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.



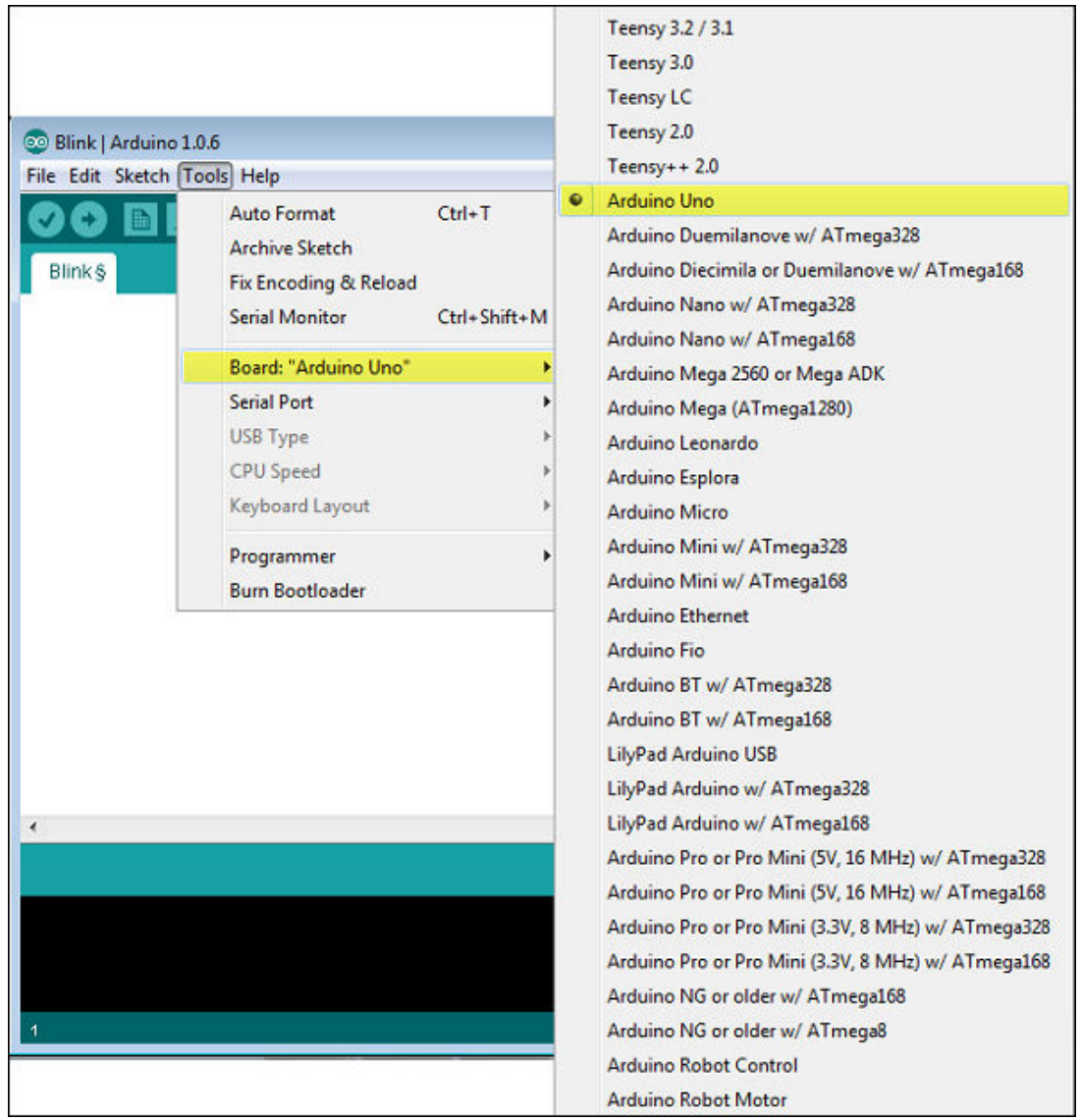
- **Step 5 – Open your first project.**
- Once the software starts, you have two options –
- Create a new project.
- Open an existing project example.
- To create a new project, select File → **New**.



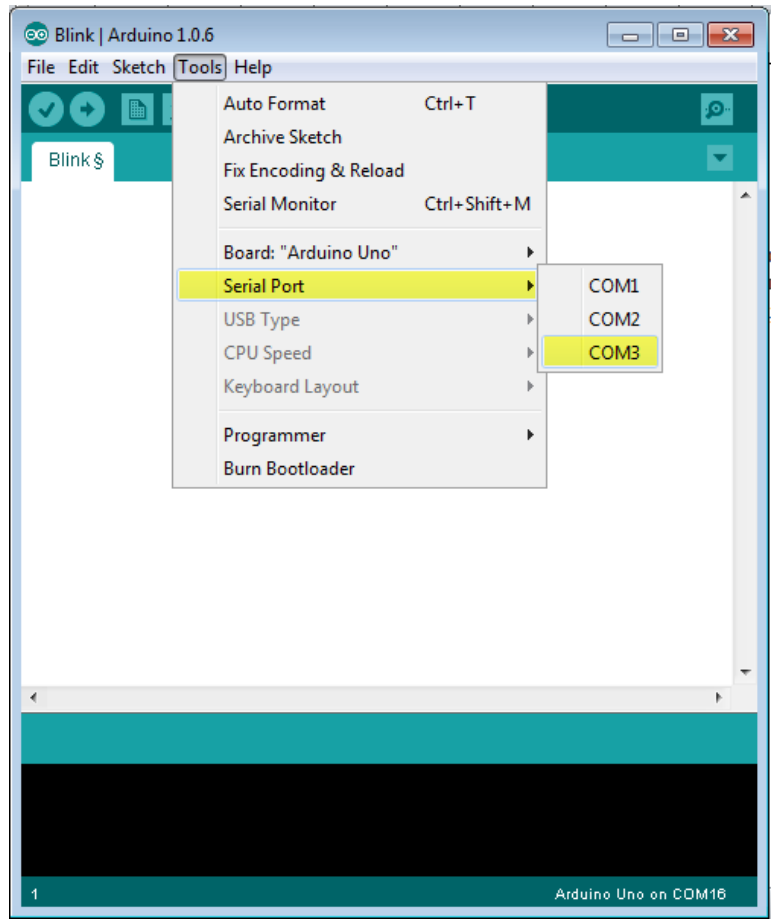
- **Step 6 – Select your Arduino board.**

To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

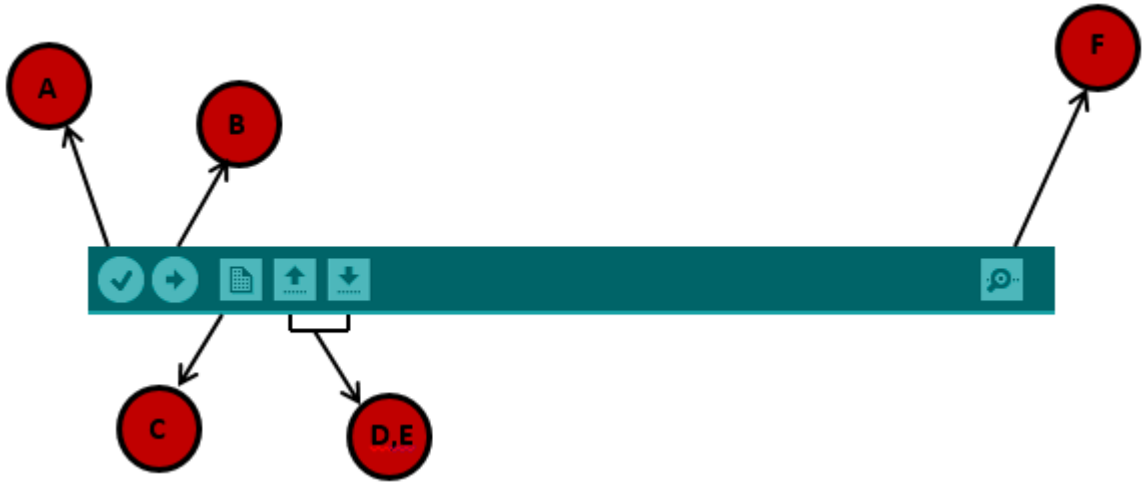
Go to Tools → Board and select your board.


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
1
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor

- **Step 7 – Select your serial port.**
- Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

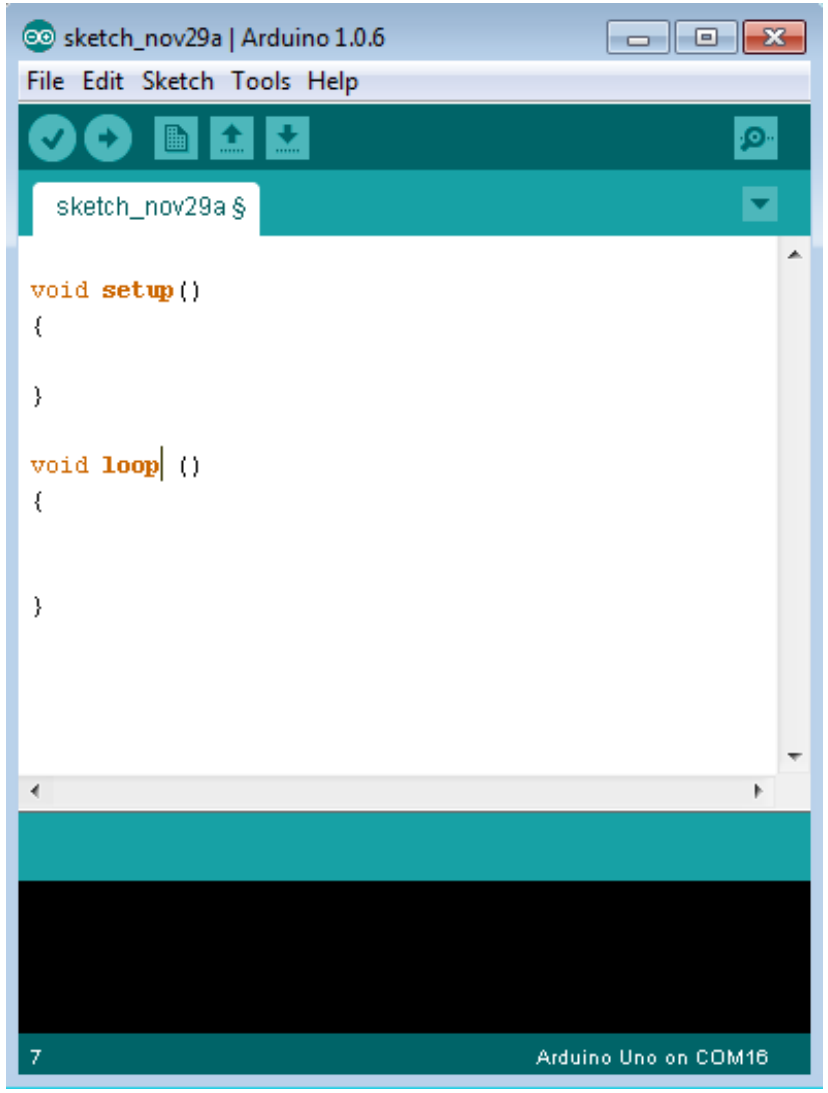

Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
COM1
COM2
COM3
1
Arduino Uno on COM16

- **Step 8 – Upload the program to your board.**
- Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.



- **A** – Used to check if there is any compilation error.
- **B** – Used to upload a program to the Arduino board.
- **C** – Shortcut used to create a new sketch.
- **D** – Used to directly open one of the example sketch.
- **E** – Used to save your sketch.
- **F** – Serial monitor used to receive serial data from the board and send the serial data to the board.

Your First Program on Arduino UNO:



Functions and Serial Communication in Arduino

CSE 315

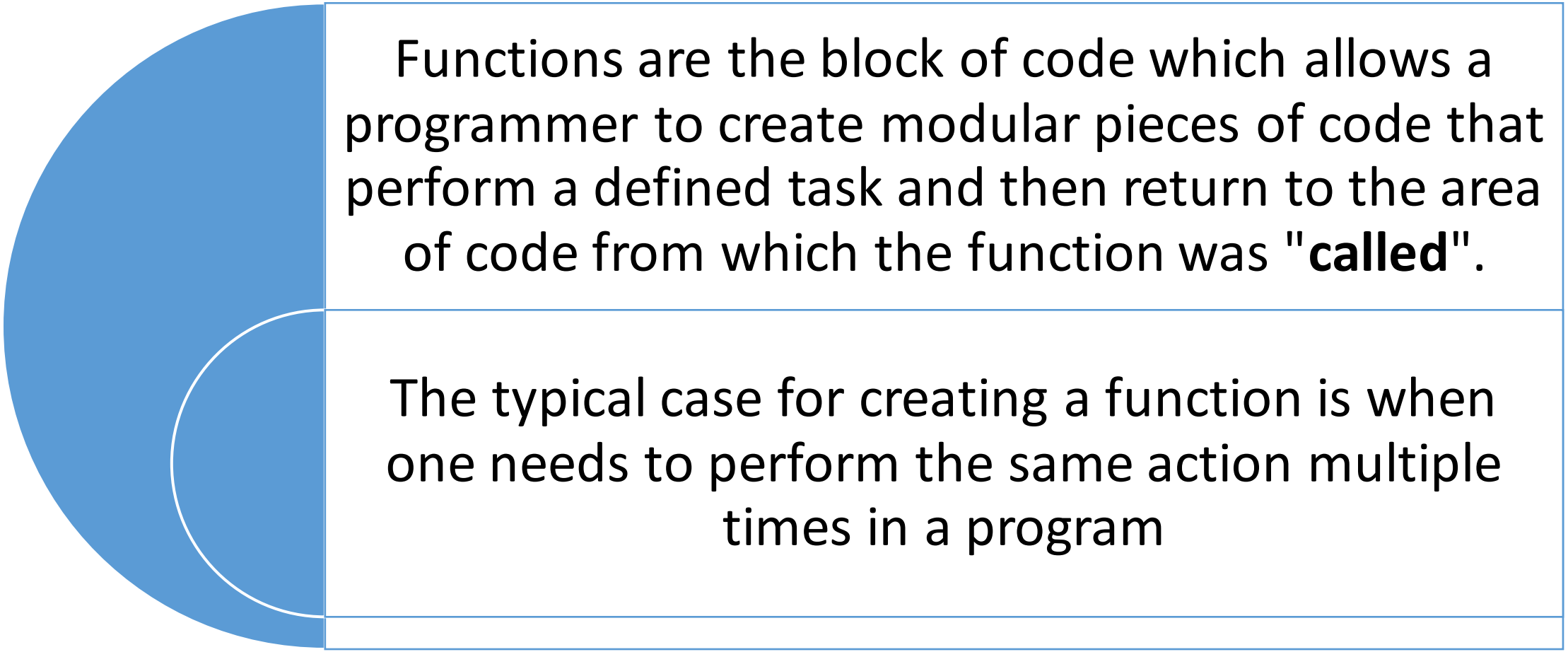
Peripherals & Interfacing

Abdullah Al Omar

Lecturer, CSE, UAP

Function

What is Function?



Advantages of Function:

Functions help the programmer stay organized. Often this helps to conceptualize the program.

Advantages of Function: (contd.)

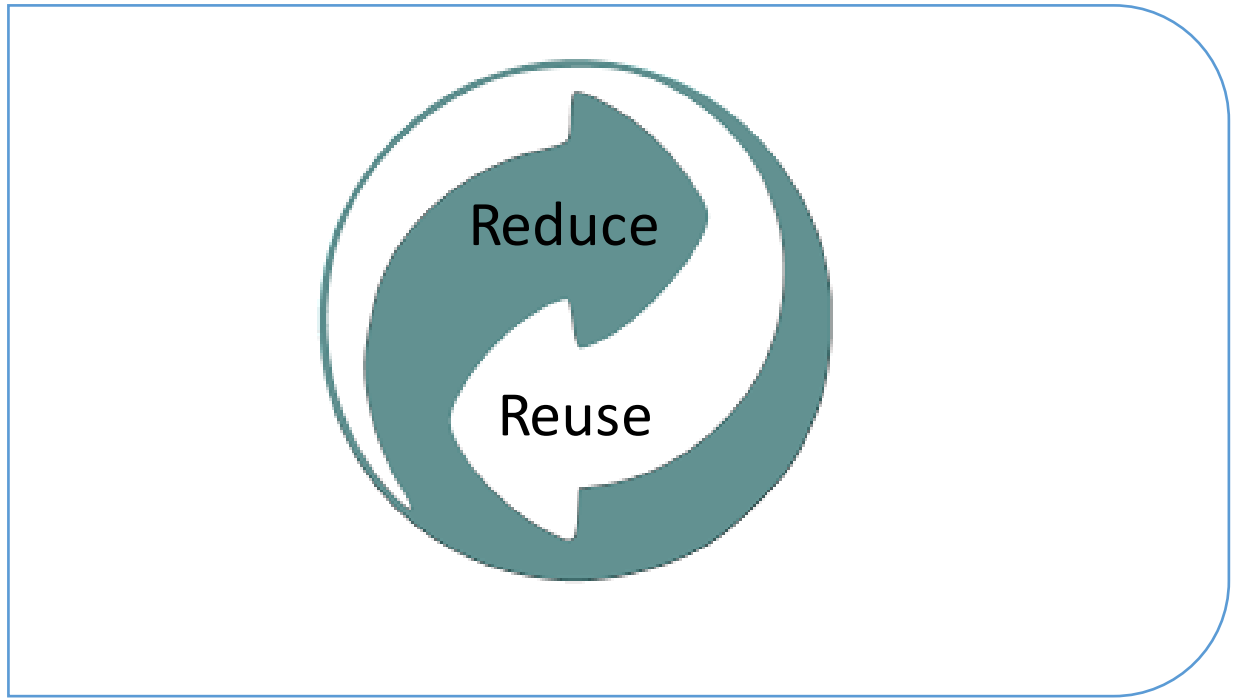
Functions codify one action in one place so that the function only has to be thought out and debugged once.

Advantages of Function: (contd.)

This also reduces chances for errors in modification, if the code needs to be changed.

Advantages of Function: (contd.)

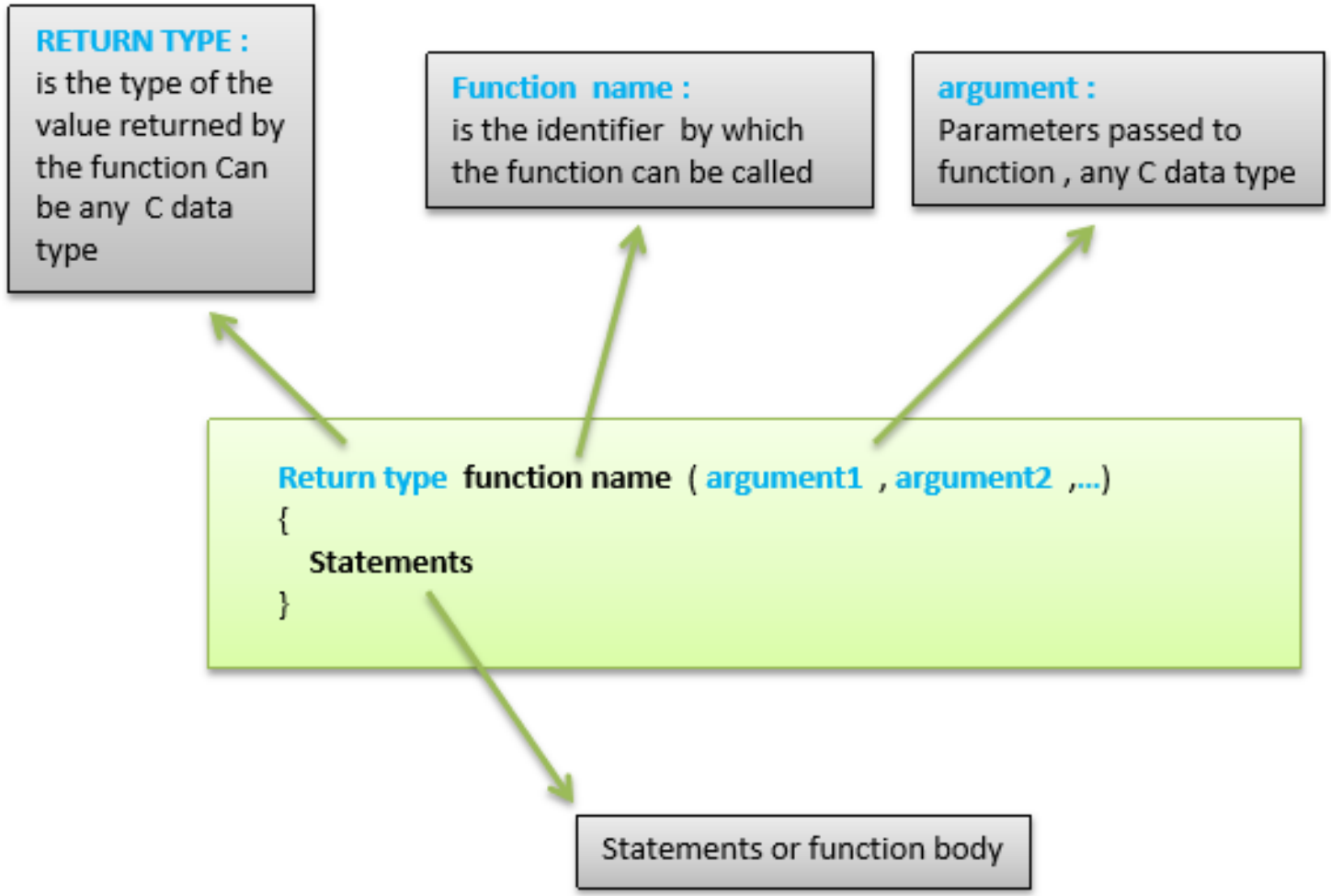
Functions make the whole sketch smaller and more compact because sections of code are reused many times.



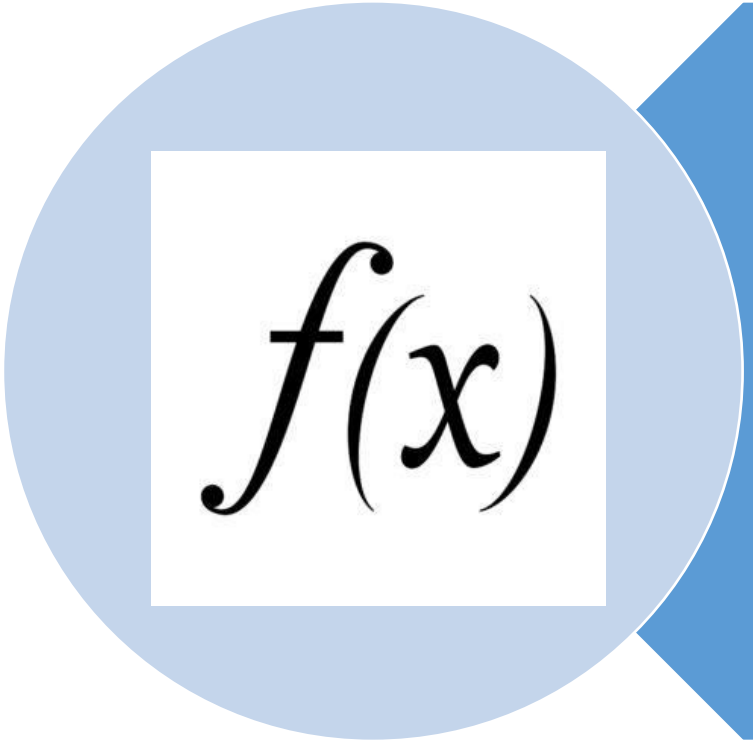
Advantages of Function: (contd.)

They make it easier to reuse code in other programs by making it more modular, and as a nice side effect, using functions also often makes the code more readable.

How Function looks like?

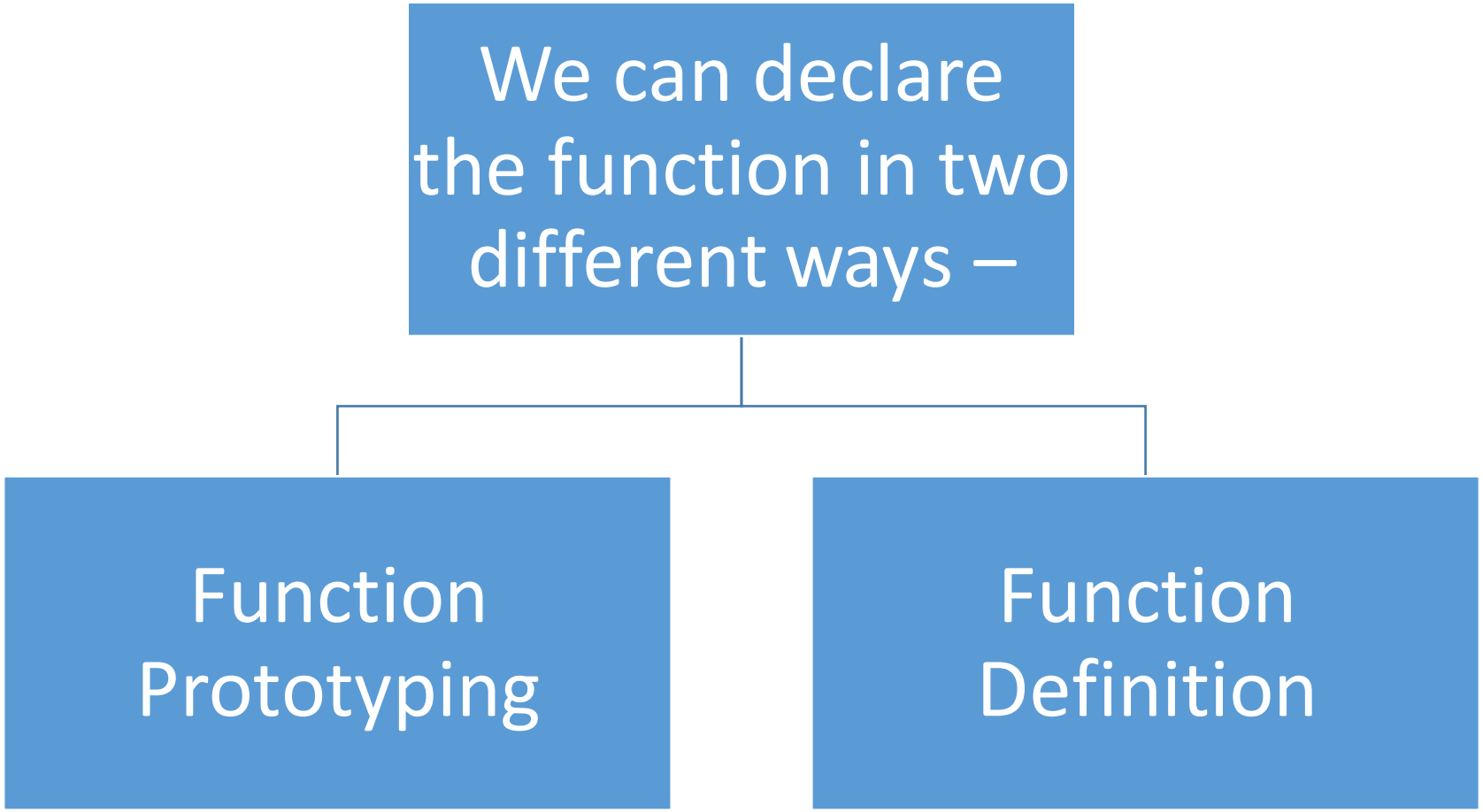


Function declaration in Arduino:



A function is declared outside any other functions, above or below the loop function.

Function declaration in Arduino: (contd.)



Function Prototyping:

The first way is just writing the part of the function called **a function prototype** above the loop function, which consists of –

- Function return type
- Function name
- Function argument type, no need to write the argument name
- Function prototype must be followed by a semicolon (;).

Function Prototyping: (contd.)

```
int sum_func (int x, int y)    // function declaration
{
        int z = 0;
        z = x+y ;
        return z; // return the value
}
void setup () {
        Statements                    // group of statements…..
}
Void loop () {
        int result = 0 ;
        result = Sum_func (5,6) ;    // function call
}
```

Function Definition:

The second part, which is called the function definition or declaration, must be declared below the loop function, which consists of –

- Function return type
- Function name
- Function argument type, here you must add the argument name
- The function body (statements inside the function executing when the function is called)

Function Definition: (contd.)

```
int sum_func (int , int ) ; // function prototype
void setup () {
        Statements // group of statements
}
Void loop () {
        int result = 0 ;
        result = Sum_func (5,6) ; // function call
}
int sum_func (int x, int y) // function declaration {
         int z = 0;
        z = x+y ;
        return z; // return the value
}
```

Class work:

Write a sketch with function to find the average of two numbers:

Serial Communication

Use of Serial:

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART), and some have several.

Serial.begin():

Sets the data rate in bits per second (baud) for serial data transmission. For communicating with Serial Monitor, make sure to use one of the baud rates listed in the menu of Arduino. You can, however, specify other rates - for example,

- to communicate over pins 0 and 1 with a component that requires a particular baud rate.

Serial.begin(): (contd.)

```
void setup() {
   Serial.begin(9600);  // opens serial port, sets data rate to 9600 bps
}

void loop() {}
```

Serial.print():

Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. Characters and strings are sent as is. For example:

- Serial.print(78) gives "78"
- Serial.print(1.23456) gives "1.23"
- Serial.print('N') gives "N"
- Serial.print("Hello world.") gives "Hello world."

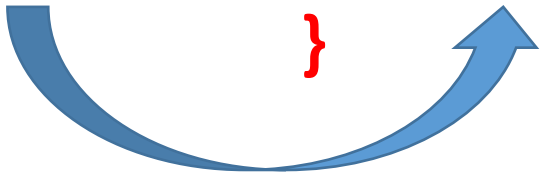
Serial.print(): [second parameter]

An optional second parameter specifies the base (format) to use; permitted values are BIN(binary, or base 2), OCT(octal, or base 8), DEC(decimal, or base 10), HEX(hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example-

- Serial.print(78, BIN) gives "1001110"
- Serial.print(78, OCT) gives "116"
- Serial.print(78, DEC) gives "78"
- Serial.print(78, HEX) gives "4E"
- Serial.print(1.23456, 0) gives "1"
- Serial.print(1.23456, 2) gives "1.23"
- Serial.print(1.23456, 4) gives "1.2346"

Printing Decimal and Binary of a number:

```
void setup() {
  Serial.begin(9600); // open the serial port at
9600 bps:
}

void loop() {
  // print labels

  Serial.print("DEC");
  Serial.print("\t");

  Serial.print("BIN");
  Serial.println();       // carriage return after
the last label

  for (int x = 0; x < 10; x++) { // only part of the ASCII chart,
change to suit
    // print it out in two formats:

    Serial.print(x, DEC);  // print as an ASCII-encoded  decimal
    Serial.print("\t");    // prints a tab


    Serial.print(x, BIN);  // print as an ASCII-encoded  binary
    Serial.println();
   delay(200);           // delay 200 milliseconds
  }
  Serial.println();       // prints another carriage return
}
```



Printing Decimal and Binary of a number: (Output)

Dec	Bin
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
.	. . . .
.	. . . .
9	1001

Serial.end()

Disables serial communication, allowing the RX and TX pins to be used for general input and output. To re-enable serial communication, call Serial.begin().

Incorporating Function and Serial:

```
int myMultiplyFunction(int x, int y){
                        int result;
                        result = x * y;
                        return result;
            }
void setup(){
            Serial.begin(9600);
}

void loop() {
            int i = 2;
            int j = 3;
             int k;

            k = myMultiplyFunction(i, j);
            Serial.println(k);   // k now contains 6

}
```

Write a sketch with function to find the average of three numbers with serial output (after decimal-“.” it will print 4 numbers):

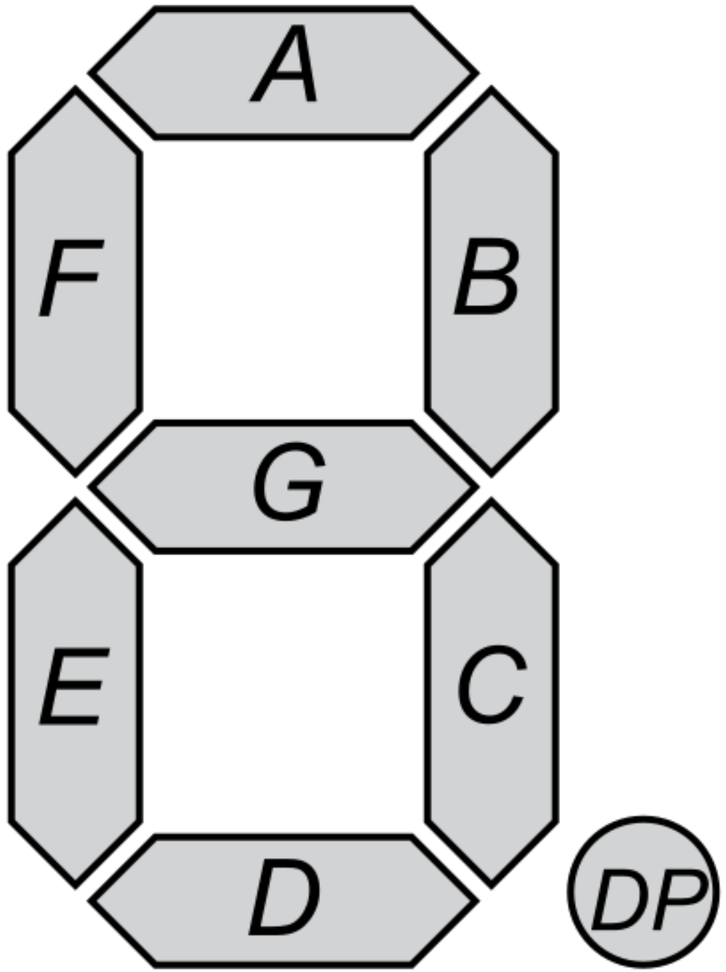
Reading Assignments:

- Serial.write()
- Serial.println()
- Serial.available()
- Serial.read()

Thank you

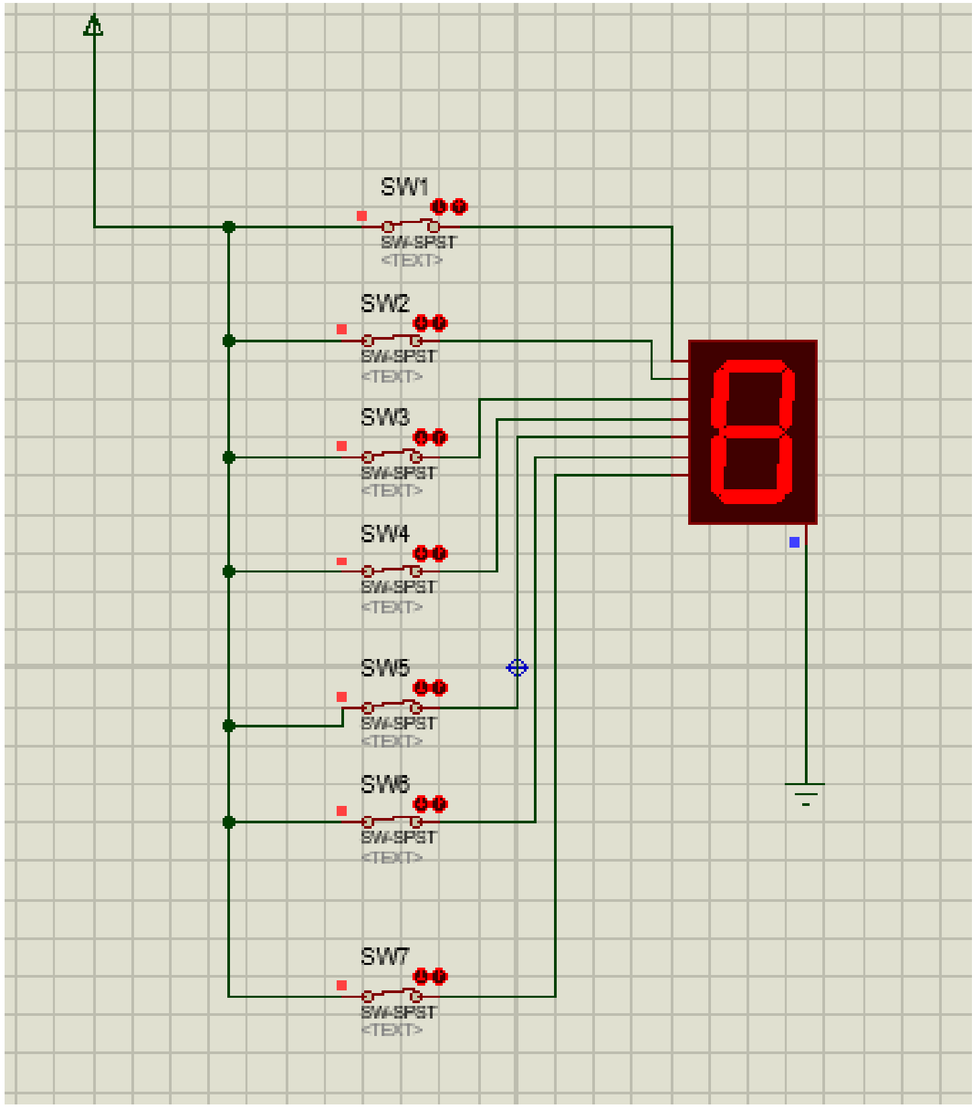
7 Segment Display

CSE 315
Peripherals & Interfacing
Abdullah Al Omar
Lecturer, CSE, UAP



Agenda

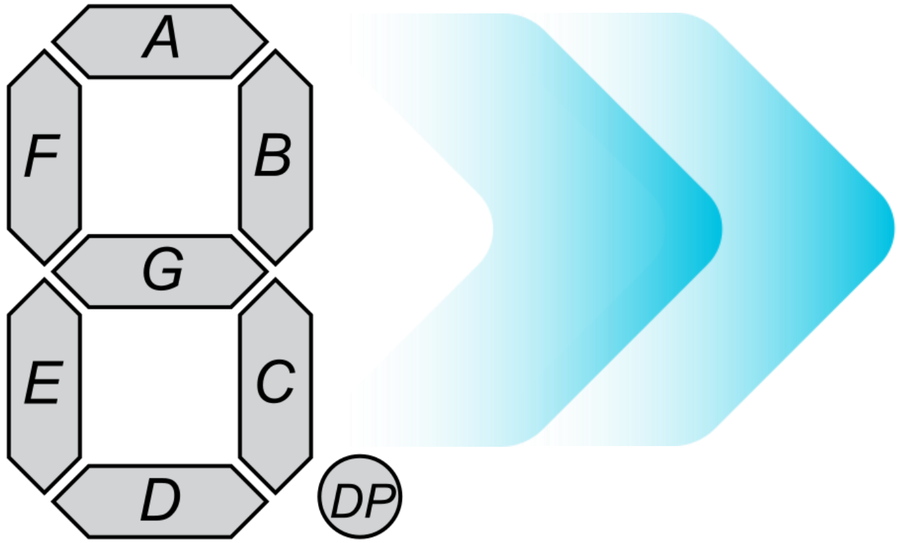
Today We'll learn about
7 segment display and
it's arduino usage



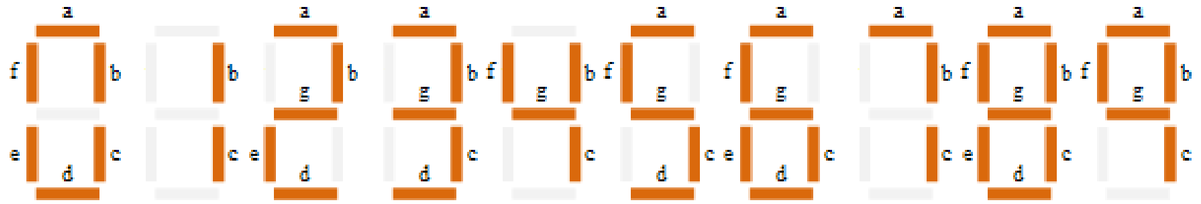
What is 7 segment display?

An LED or Light Emitting Diode, is a solid state optical pn-junction diode which emits light energy in the form of photon. The main advantage of light emitting diodes is that because of their small die size, several of them can be connected together within one small and compact package producing what is generally called a 7-segment Display.

Why 7 Segment?



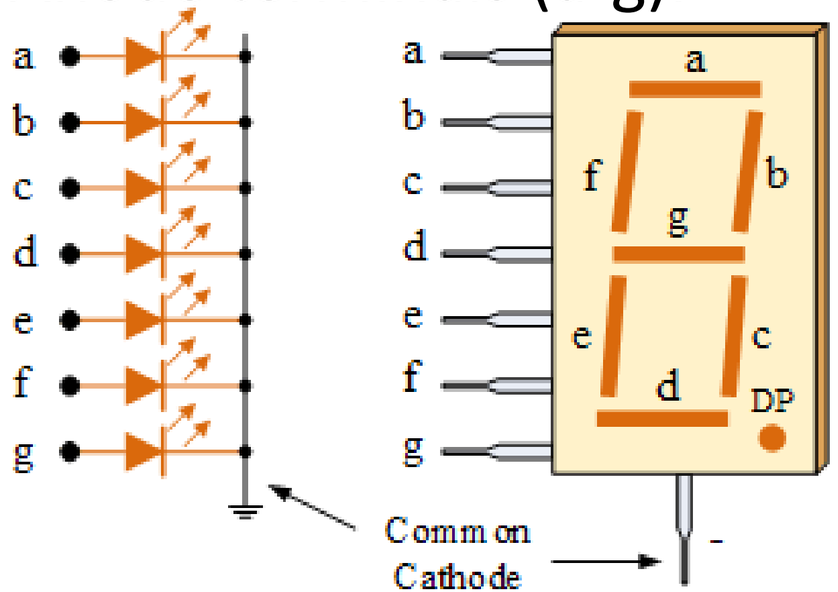
7-Segment Display Segments for all Numbers.



Then for a 7-segment display, we can produce a truth table giving the individual segments that need to be illuminated in order to produce the required decimal digit from 0 through 9 as shown below.

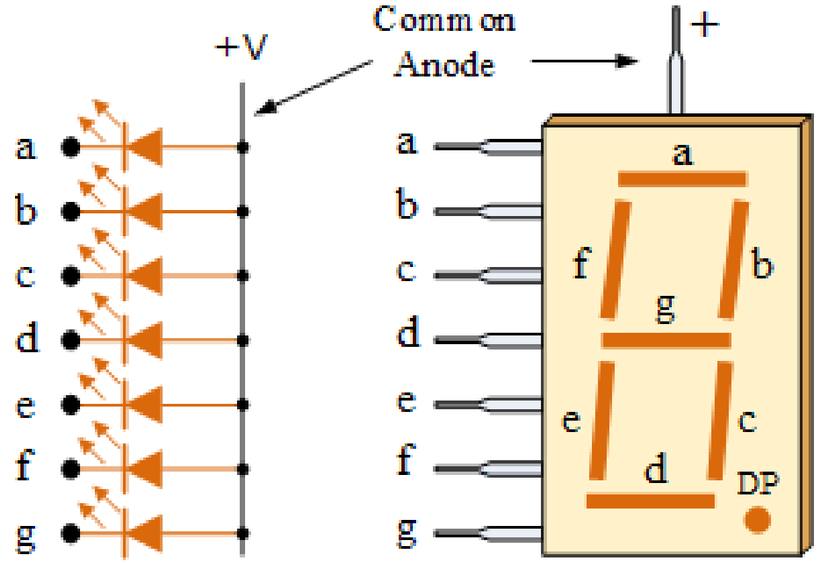
Types of 7 Segment display

- The Common Cathode (CC) – In the common cathode display, all the cathode connections of the LED segments are joined together to logic “0” or ground. The individual segments are illuminated by application of a “HIGH”, or logic “1” signal via a current limiting resistor to forward bias the individual Anode terminals (a-g).



Types of 7 segment display (contd.)

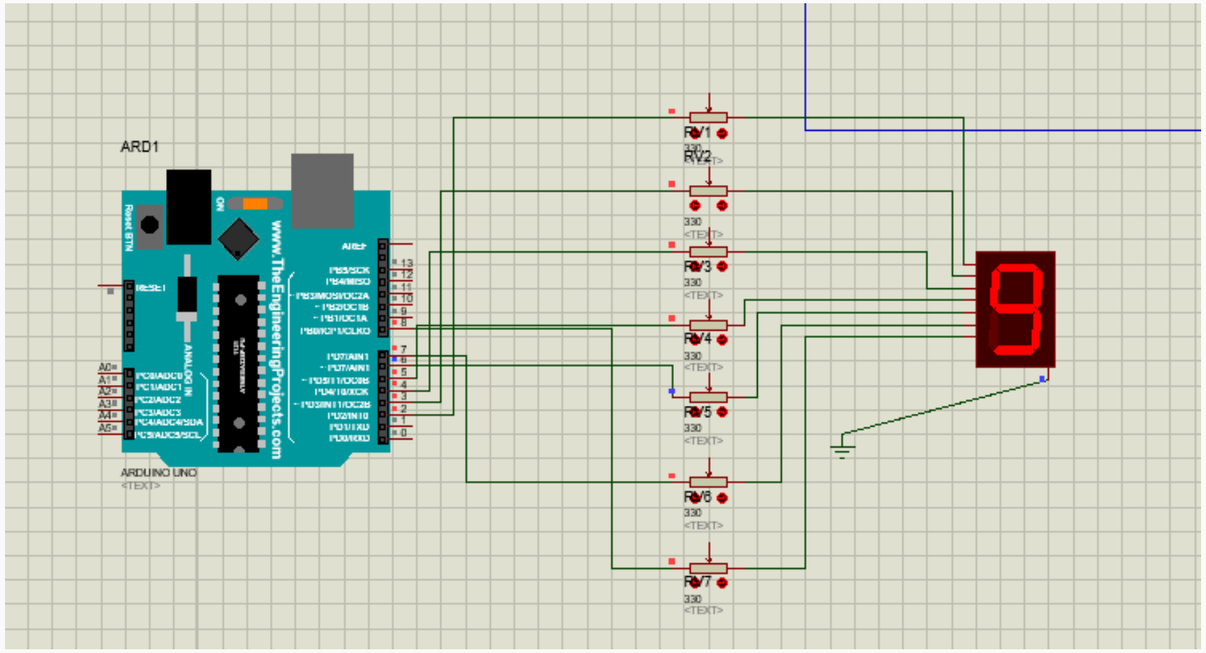
- The Common Anode (CA) – In the common anode display, all the anode connections of the LED segments are joined together to logic “1”. The individual segments are illuminated by applying a ground, logic “0” or “LOW” signal via a suitable current limiting resistor to the Cathode of the particular segment (a-g).



Truth Table of 7 segment

7-segment Display Truth Table

Decimal Digit	Individual Segments Illuminated						
	a	b	c	d	e	f	g
0	×	×	×	×	×	×	
1		×	×				
2	×	×		×	×		×
3	×	×	×	×			×
4		×	×			×	×
5	×		×	×		×	×
6	×		×	×	×	×	×
7	×	×	×				
8	×	×	×	×	×	×	×
9	×	×	×			×	×



Here "x" means switch is closed/connected so that current can pass through that pin

Some Use Case

Counte
r

Speed measuring
device

Digital Clock

Cashier machine

Digital
meter

Microwave
oven

Washing machine

Lets see codes & Simulation output

```
void setup() {
  // put your setup code here, to run once:
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
zero();
one();
two();
three();
four();
five();
six();
seven();
eight();
nine();
}
```

```
void zero() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
  delay(1000);
}
void one(){
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  delay(1000);

}
```

```
void two() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
  delay(1000);
}
```

```
 void three() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
  delay(1000);

 }
```

```
void four() {
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);

}
```

```
void five() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);

}
```

```
void six() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}
```

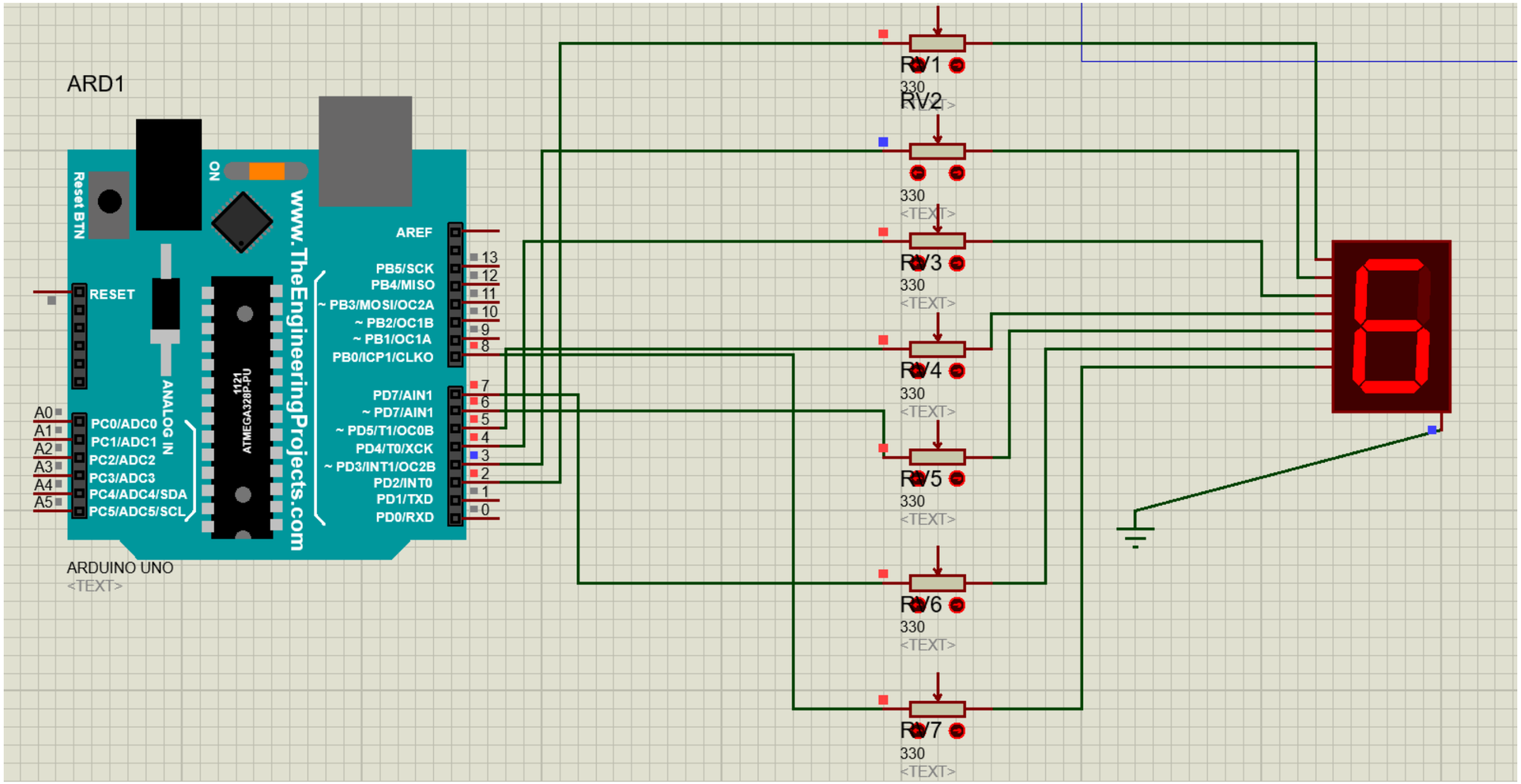
```
void seven() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  delay(1000);
  }
```

```
void eight() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);

}

void nine() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}
```

Equipments:

- Arduino Uno
- 7SEG-COM-CATHODE
- Resistor 330 Ohm (X7)
- Ground

Arduino Design:



4 digit 7 segment:

7seg4digit

```
void setup() {
  // put your setup code here, to run once:
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
pinMode(9,OUTPUT);
pinMode(10,OUTPUT);
pinMode(11,OUTPUT);
pinMode(12,OUTPUT);
pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(9,HIGH);
digitalWrite(10,HIGH);
digitalWrite(11,HIGH);
digitalWrite(12,HIGH);
zero();
one();
two();
three();
four();
five();
six();
seven();
eight();
nine();
}
```

7seg4digit

```
void zero() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,LOW);
  delay(1000);
}
void one(){
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  delay(1000);

}

void two() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
  delay(1000);
}
```

7seg4digit

```
void three() {
 digitalWrite(2,HIGH);
 digitalWrite(3,HIGH);
 digitalWrite(4,HIGH);
 digitalWrite(5,HIGH);
 digitalWrite(6,LOW);
 digitalWrite(7,LOW);
 digitalWrite(8,HIGH);
 delay(1000);

}

void four() {
  digitalWrite(2,LOW);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);

}

void five() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
```

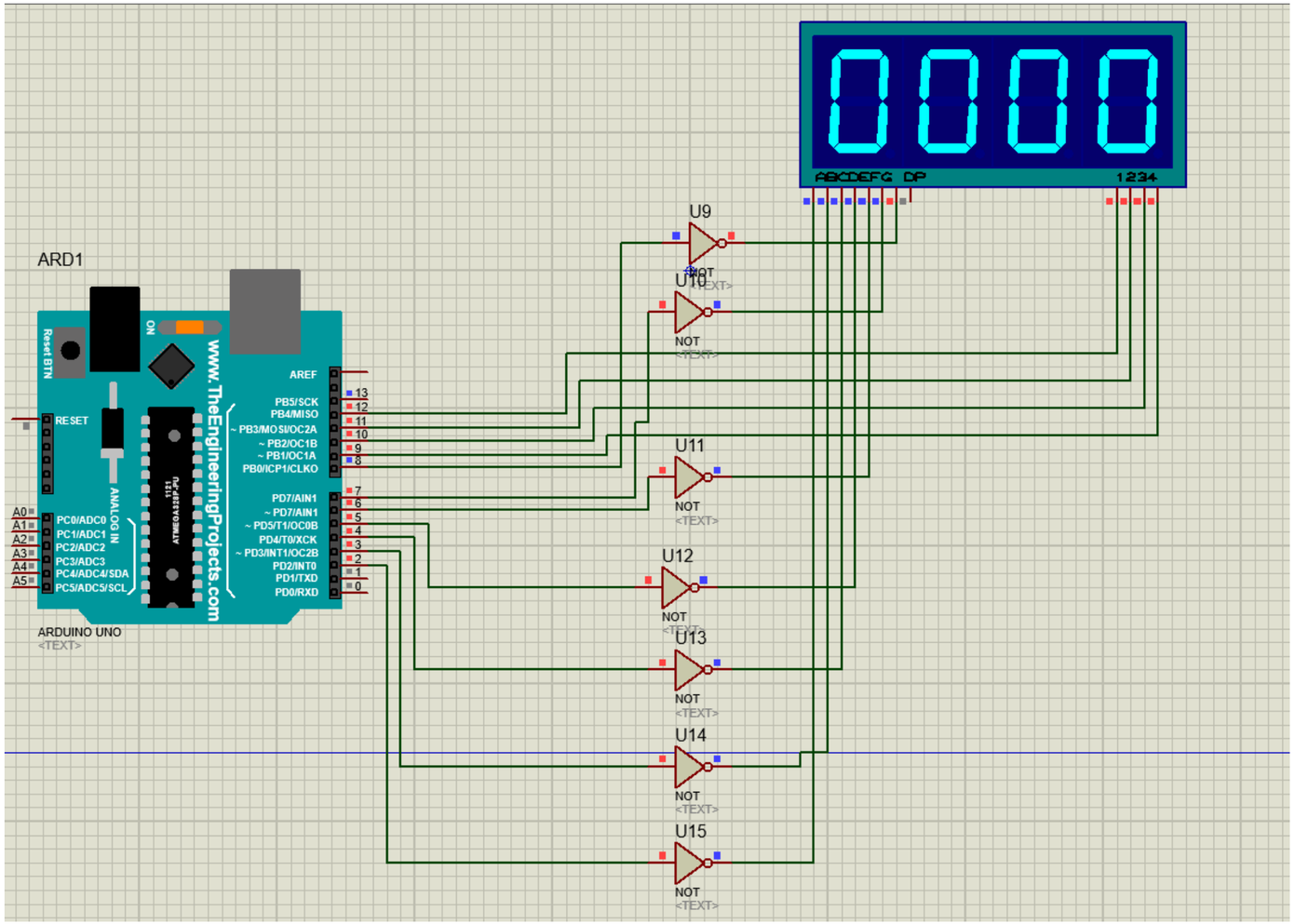
```
void six() {
  digitalWrite(2,HIGH);
  digitalWrite(3,LOW);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}

void seven() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
  digitalWrite(8,LOW);
  delay(1000);
  }

void eight() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,HIGH);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);

}
```

```
void nine() {
  digitalWrite(2,HIGH);
  digitalWrite(3,HIGH);
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH);
  digitalWrite(6,LOW);
  digitalWrite(7,HIGH);
  digitalWrite(8,HIGH);
  delay(1000);
}
```

Equipments:

- Arduino Uno
- 7SEG-MPX4-CA-BLUE
- Resistor (X7)
- NOT(X7)

4 digit 7 segment:



Thank you!

Arithmetic logic unit (ALU)

Name : Sathy Akter

ID: 18101077

Content

- What is ALU?
- Introducing to Arithmetic Unit and Logical Unit.
- How does occur Binary Addition.
- How does occur Binary Subtraction.
- Circuit Diagram for ALU.
- Describing about Code.

Definition of ALU

In computing, an arithmetic logic unit is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers. This is in contrast to a floating-point unit, which operates on floating point numbers.

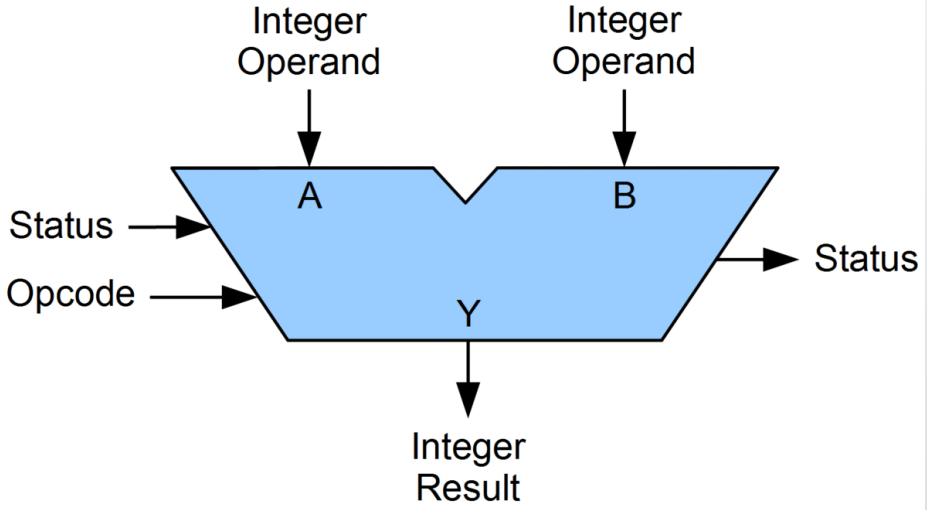
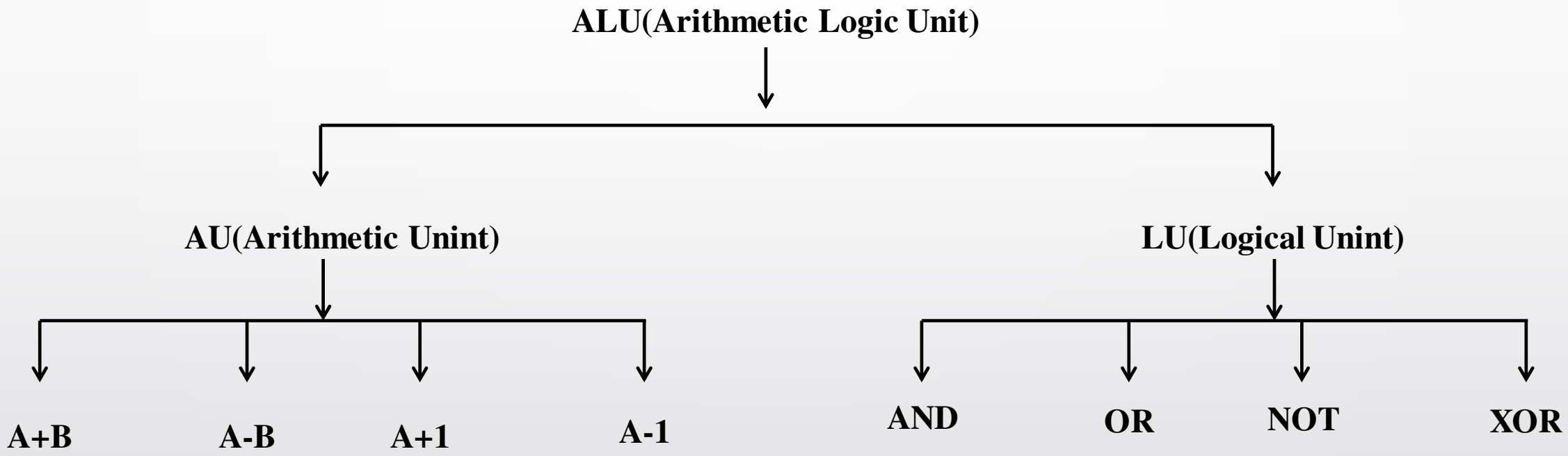


Fig: ALU

Image Source: www.google.com

What is the basic use of ALU unit?

An **arithmetic logic unit** (**ALU**), a memory **unit**, and input/output (I/O) controllers. The **ALU** performs **simple** addition, subtraction, multiplication, division, and logic operations, such as OR and AND. The memory stores the program's instructions and data.

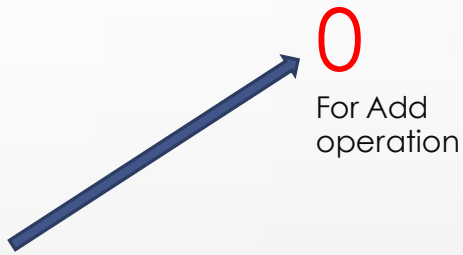

ALU(Arithmetic Logic Unit)
AU(Arithmetic Unint)
LU(Logical Unint)
A+B
A-B
A+1
A-1
AND
OR
NOT
XOR

Let's make 2 bit **ALU** with Arduino.

Binary ADD:

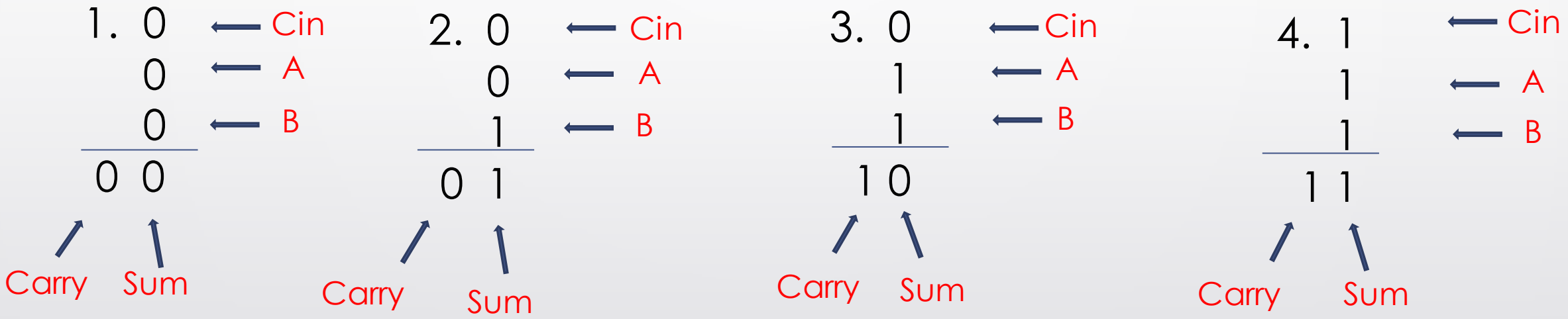
C_{in} =	1	0	1	1		1	1	0	
A =	1	1	0	1		1	0	1	1 (219)
+ B =	1	1	0	0		0	1	1	0 (198)
	1	1	0	1	0	0	0	0	1 (417)

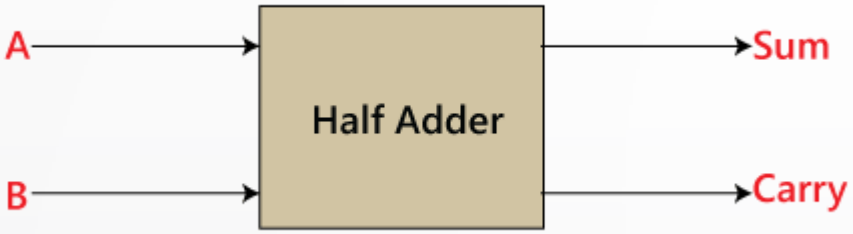
Binary ADD:

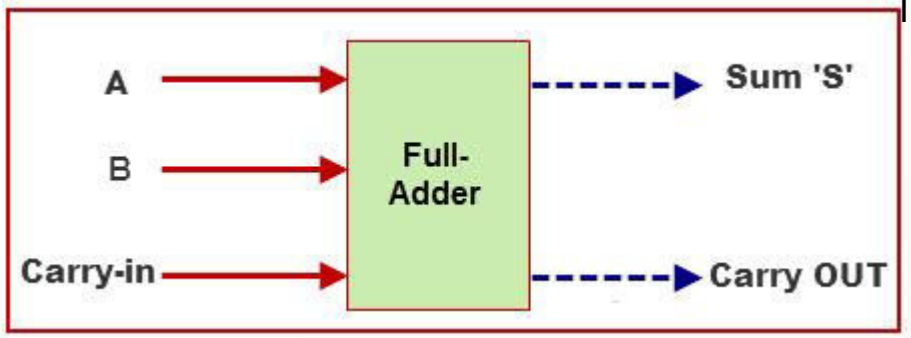


C_{in} =	1	0	1	1	1	1	0		
A =	1	1	0	1	1	0	1	1	(219)
+ B =	1	1	0	0	0	1	1	0	(198)
	1	1	0	1	0	0	0	0	1 (417)

Binary addition bit combination:







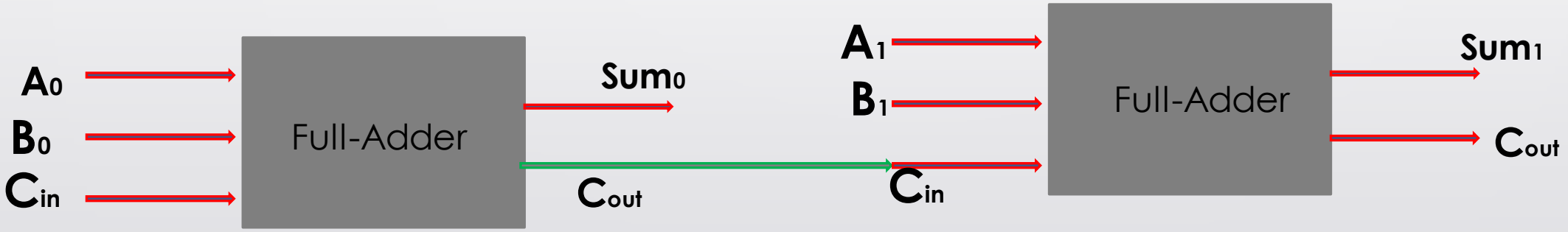


Fig: 2 bit addition

Image Source: www.google.com

Binary Subtraction:

A = 1 1 0 1 1 0 1 1 (219)

B = 1 1 0 0 0 1 1 0 (198)

B = 0 0 1 1 1 0 0 1 [1's complement]

+ 1

B'= 0 0 1 1 1 0 1 0 [2's complement]

0

For Add operation

C_{in} = 1 1 1 1 0 1 0

A = 1 1 0 1 1 0 1 1 (219)

+ B' = 0 0 1 1 1 0 1 0 (198)

1 0 0 0 1 0 1 0 1 (21)

ALU Truth Table:

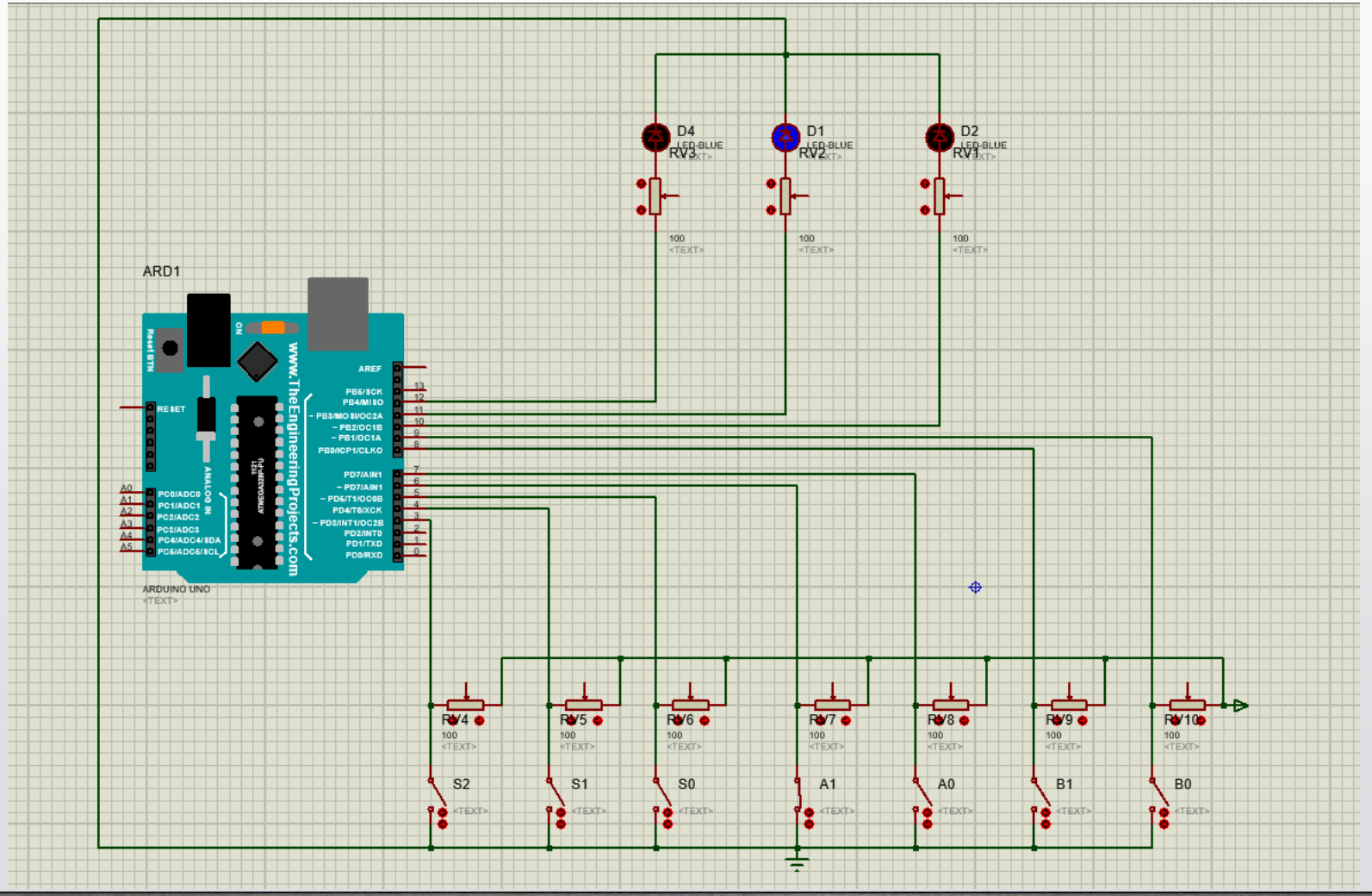
Selection Input			Operation Performed
0	0	0	A + B
0	0	1	A - B
0	1	0	A - 1
0	1	1	A + 1
1	0	0	A and B
1	0	1	A or B
1	1	0	not A
1	1	1	A xor B

Image Source: www.google.com

Arduino Input, Output Pin number:

```
//SWITCH
pinMode(5, INPUT); //S0
pinMode(4, INPUT); //S1
pinMode(3, INPUT); //S2

//INPUT
pinMode(7, INPUT); //A[0]
pinMode(6, INPUT); //A[1]
pinMode(9, INPUT); //B[0]
pinMode(8, INPUT); //B[1]

//OUTPUT
pinMode(10, OUTPUT); //S[0]
pinMode(11, OUTPUT); //S[1]
pinMode(12, OUTPUT); //COUT
```

Circuit implementation using Proteus:



Functions I made for make 2 bit ALU.

```
void FULLADDER()
{
  COUT = LOW;
  for (int i = 0; i <= 1; i++)
  {
    int COUNT = 0;
    if (A[i] == HIGH)
    {
      COUNT = COUNT + 1;
      if (B[i] == HIGH)
      {
        COUNT = COUNT + 1;
        if (COUT == HIGH)
        {
          COUNT = COUNT + 1;
        }
        else if (COUT == LOW)
        {
          COUNT = COUNT + 0;
        }
      }
      else if (B[i] == LOW)
      {
        COUNT = COUNT + 0;
        if (COUT == HIGH)
        {
          COUNT = COUNT + 1;
        }
        else if (COUT == LOW)
        {
          COUNT = COUNT + 0;
        }
      }
    }
    else if (A[i] == LOW)
    {
      COUNT = COUNT + 0;
      if (B[i] == HIGH)
      {
        COUNT = COUNT + 1;
        if (COUT == HIGH)
        {
          COUNT = COUNT + 1;
        }
        else if (COUT == LOW)
        {
          COUNT = COUNT + 0;
        }
      }
      else if (B[i] == LOW)
      {
        COUNT = COUNT + 0;
        if (COUT == HIGH)
        {
          COUNT = COUNT + 1;
        }
        else if (COUT == LOW)
        {
          COUNT = COUNT + 0;
        }
      }
    }
    if (COUNT == 3)
    {
      S[i] = HIGH;
      COUT = HIGH;
    }
    else if (COUNT == 2)
    {
      S[i] = LOW;
      COUT = HIGH;
    }
    else if (COUNT == 1)
    {
      S[i] = HIGH;
      COUT = LOW;
    }
    else if (COUNT == 0)
    {
      S[i] = LOW;
      COUT = LOW;
    }
  }
}
```

```
void ADD_1(int *X)
{
  if (*(X + 0) == HIGH)
  {
    //Carry=1
    *(X + 0) = LOW;
    if (*(X + 1) == HIGH)
    {
      *(X + 1) = LOW;
      COUT = HIGH;
    }
    else if (*(X + 1) == LOW)
    {
      *(X + 1) = HIGH;
      COUT = LOW;
    }
  }
  else if (*(X + 0) == LOW)
  {
    //Carry=0
    *(X + 0) = HIGH;
    if (*(X + 1) == HIGH)
    {
      *(X + 1) = HIGH;
      COUT = LOW;
    }
    else if (*(X + 1) == LOW)
    {
      *(X + 1) = LOW;
      COUT = LOW;
    }
  }
}
```

```
void NOT(int *X)
{
  for (int i = 0; i <= 1; i++)
  {
    if (*(X + i) == HIGH)
    {
      *(X + i) = LOW;
    }
    else
    {
      *(X + i) = HIGH;
    }
  }
  COUT = LOW;
}
void AND()
{
  for (int i = 0; i <= 1; i++)
  {
    if (A[i] == HIGH && B[i] == HIGH)
    {
      S[i] = HIGH;
    }
    else
    {
      S[i] = LOW;
    }
  }
  COUT = LOW;
}
```

```
void OR()
{
  for (int i = 0; i <= 1; i++)
  {
    if (A[i] == LOW && B[i] == LOW)
    {
      S[i] = LOW;
    }
    else
    {
      S[i] = HIGH;
    }
  }
  COUT = LOW;
}
```

```
void XOR()
{
  for (int i = 0; i <= 1; i++)
  {
    if ((A[i] == LOW && B[i] == LOW) || (A[i] == HIGH && B[i] == HIGH))
    {
      S[i] = LOW;
    }
    else
    {
      S[i] = HIGH;
    }
  }
  COUT = LOW;
}
```

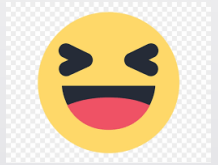
Increment(A+1)

Let's assume A=10 (2)

And B=1 (1)
Or we can represent it as B=01 (1)

1 0 (A)
0 1 (B)
---
0 1 1

Okay I am done, here is your result



Calling Fulladder

0 1 1

Image Source: www.google.com

Decrement(A-1)

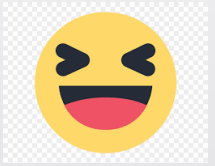
Let's assume A=10 (1)

And B=1
Or we can represent it as B=01
After 1's complement B'=10
After 2's complement B''=11

Calling Fulladder

$$\begin{array}{r} 1\ 0 \\ 1\ 1 \\ \hline 1\ 0\ 1 \end{array}$$

Okay I am done, here is your result



1 0 1

Image Source: www.google.com

Thank You

Image Source: www.google.com

2 bits ALU

Peripheral and Interfacing

Presented by:

Nuzhat Tabassum Progga

ID: 18101005

What is ALU?

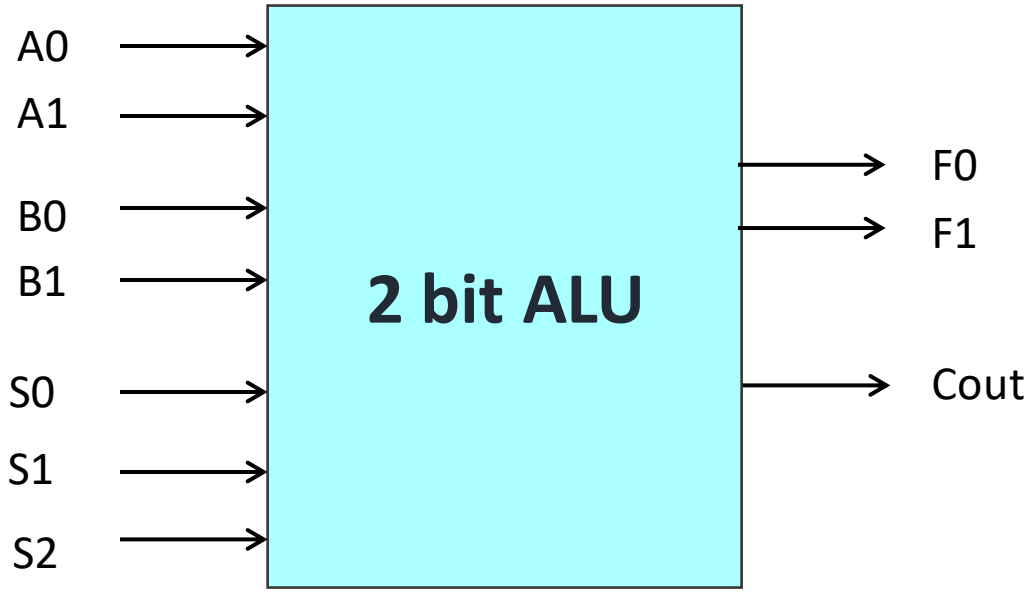
ALU (Arithmetic and Logic Unit) is the part of computer processor that is used for doing arithmetic or logic operation on operands.

What is the basic use of ALU unit?

An arithmetic logic unit (ALU), a memory unit, and input/output(I/O) controllers. The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND. The memory stores the program's instructions and data.

2 bit ALU block diagram

Here,
A(A1 A0) = operand
B(B1 B0) = operand
S(S2 S1 S0) = Select inputs
F(F1 F0) = Output
Cout = Carry output



Truth Table:

S2	S1	S0	Output	Function
0	0	0	F = A	Transfer A
0	0	1	F = A + 1	Increment A
0	1	0	F = A+B	Add B to A
0	1	1	F = A+B+1	Add B to A plus 1
1	0	0	F = A + $\overline{B}$	Add 1's compliment of B to A
1	0	1	F = A + $\overline{B} + 1$	Add 2's compliment of B to A
1	1	0	F = A-1	Decrement A
1	1	1	F = A	Transfer A

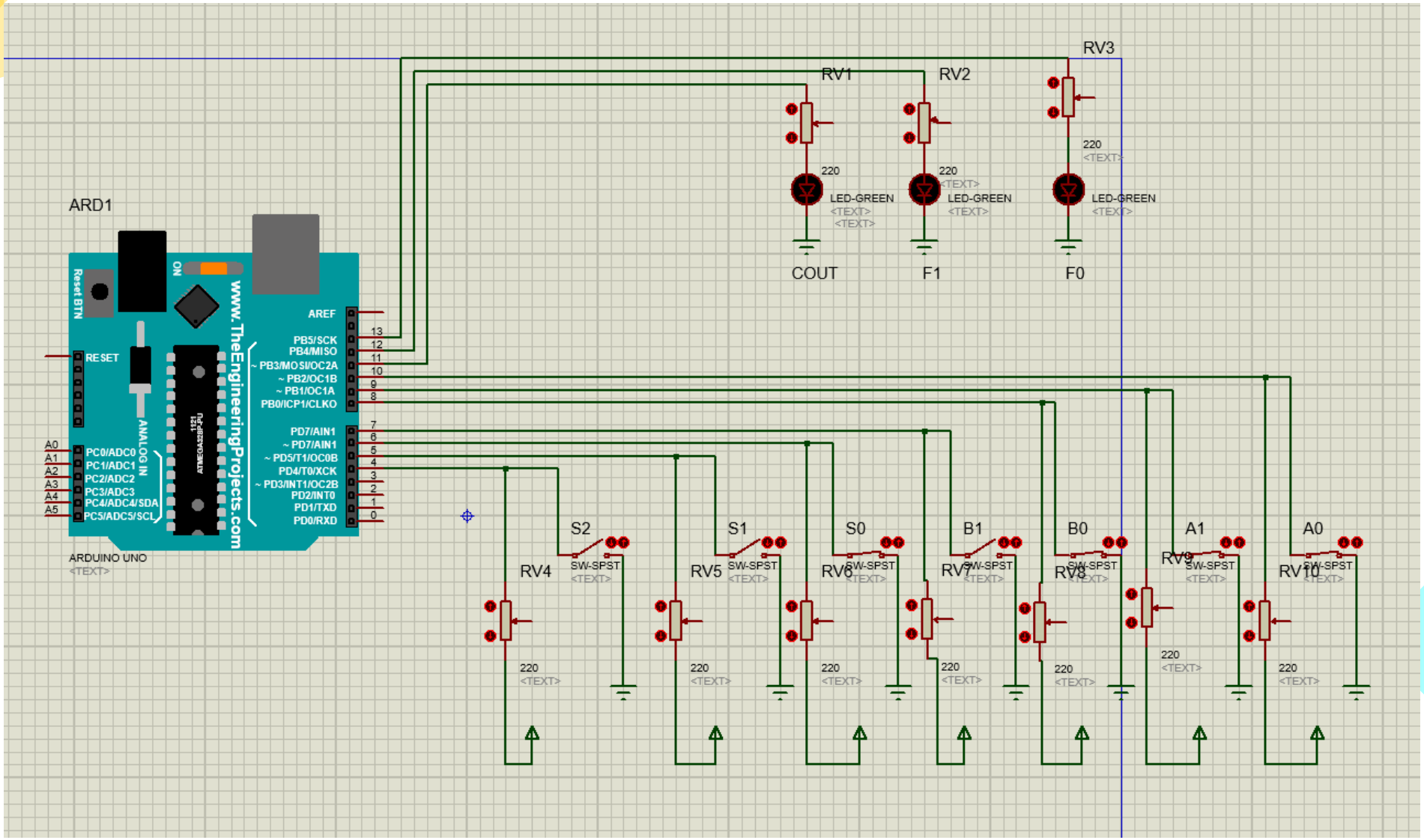
Now we a going to design the circuit using Proteus. To do so we have to use some pins. We will use digital pin :

Pin num	Type	Purpose
13	Output	F0
12	Output	F1
11	Output	Cout
10	Input	A0
9	Input	A1
8	Input	B0
7	Input	B1
6	Input	S0
5	Input	S1
4	Input	S2

To design this circuit on proteus we will need:

1. Arduino UNO
2. LED
3. Resistor
4. Switch SPST
5. Power
6. ground

Circuit implementation using Proteus



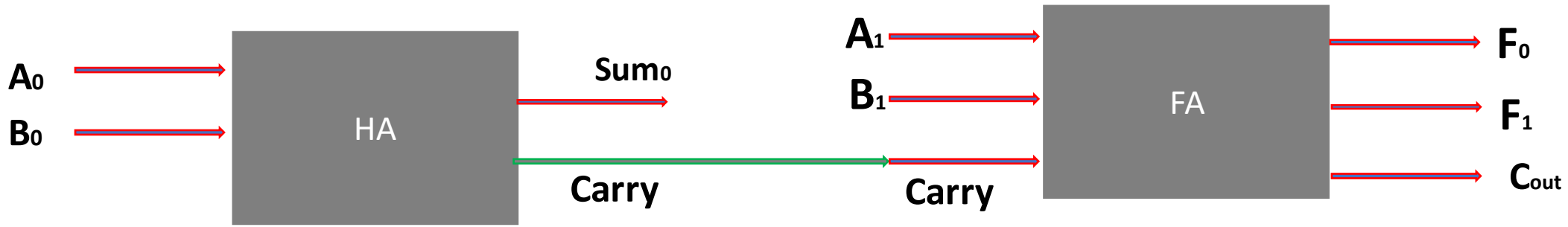
Basic Knowledge we will need to implement the code:

1. Calculate sum and carry with half adder:
   sum = $A \oplus B$
   carry = AB

2. Calculate sum and carry with full adder:
   sum = sum = $A \oplus B \oplus C$
   carry = $AB + BC + CA$

3. Calculate difference and borrow with half subtractor:
   difference = $A \oplus B$
   borrow = $\bar{A}B$

4. Calculate difference and borrow with full subtractor:
   difference = $A \oplus B \oplus C$
   borrow = $BC + \bar{A}(B + C)$


A0
B0
HA
Sum0
Carry
A1
B1
Carry
FA
F0
F1
Cout

Some operation:

1. For S2= 0 S1 = 1 S0 = 0 out operation will be A+B
   let, A = 1 1 & B = 1 1

Carry =	1	1	
A =		1	1
B =		1	1
	1	1	0

Here,
For A0 and B0 we use half adder:
Sum (HA) = A0 $\oplus$ B0 = 1 $\oplus$ 1=0
Carry (HA)=A0.B0=1.1=1

For A1 and B1 we use full adder , because we get a carry
Sum (FA) = A1 $\oplus$ B1 $\oplus$ Carry = 1 $\oplus$ 1 $\oplus$ 1=1
Cout (FA) = A1.B1 + B1.Carry + Carry.A1= 1.1 +1.1+1.1=1

Some operation:

2. For S2= 1 S1 = 0 S0 = 1 out operation will be $\mathbf{A + \overline{B} + 1}$
   let, A = 1 1, $\overline{\mathbf{B}}$ = 1 0

Carry =	1	1	
A =		1	1
B =		1	0
			1
	1	1	0

Here,
For A0 and B0 we use full adder:
Sum (HA) = A0 $\oplus$ B0 $\oplus$ 1= 1 $\oplus$ 0 $\oplus$ 1=0
Carry (HA)=A0.B0 + B0.1 + 1.A0 = 1.0 + 0.1 + 1.1 = 1

For A1 and B1 we use full adder , because we get a carry
Sum (FA) = A1 $\oplus$ B1 $\oplus$ Carry = 1 $\oplus$ 1 $\oplus$ 1 = 1
Cout (FA) = A1.B1 + B1.Carry + Carry.A1 = 1.1+1.1+1.1 =1

Code:

```
int a0, a1, b0, b1, c_out, s0, s1, s2, f0, f1;
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(11,OUTPUT);

  pinMode(10,INPUT);
  pinMode(9,INPUT);
  pinMode(8,INPUT);
  pinMode(7,INPUT);
  pinMode(6,INPUT);
  pinMode(5,INPUT);
  pinMode(4,INPUT);


}
```

```
void loop() {
  a0 = digitalRead(10);
  a1 = digitalRead(9);

  b0 = digitalRead(8);
  b1 = digitalRead(7);

  s0 = digitalRead(6);//s0
  s1 = digitalRead(5);//s1
  s2 = digitalRead(4);//s2
  if((s2 == 0 && s1 == 0 && s0 == 0)||(s2 == 1 && s1 == 1 && s0 == 1)){
    c_out = 0;
    digitalWrite(13,a0);
    digitalWrite(12,a1);
    digitalWrite(11,c_out);

  }
  else if(s2 == 0  && s1 == 0 && s0 == 1){

    f0 = a0 ^ 1;
    int cin = a0 & 1;
    f1 = cin ^ a1;
    c_out = cin & a1;
    digitalWrite(13,f0);
    digitalWrite(12,f1);
    digitalWrite(11,c_out);

  }
```

```
else if(s2 == 0  && s1 == 1 && s0 == 0){
  f0 = a0 ^ b0;
  int cin = a0 & b0;
  f1 = cin ^ a1 ^ b1;
  c_out = (cin & a1)|(b1 & a1)|(cin & b1);
  digitalWrite(13,f0);
  digitalWrite(12,f1);
  digitalWrite(11,c_out);

}else if(s2 == 0  && s1 == 1 && s0 == 1){
  int p = 1;
  f0 = a0 ^ b0 ^ p;
  int cin = (a0 & b0)|(p & b0)|(a0 & p);
  f1 = cin ^ a1 ^ b1;
  c_out = (cin & a1)|(b1 & a1)|(cin & b1);
  digitalWrite(13,f0);
  digitalWrite(12,f1);
  digitalWrite(11,c_out);

}
```

```
else if(s2 == 1  && s1 == 0 && s0 == 0){
  b0 = !b0;
  b1 = !b1;
  f0 = a0 ^ b0 ;
  int cin = a0 & b0;
  f1 = cin ^ a1 ^ b1;
  c_out = (cin & a1)|(b1 & a1)|(cin & b1);
  digitalWrite(13,f0);
  digitalWrite(12,f1);
  digitalWrite(11,c_out);
}else if(s2 == 1  && s1 == 0 && s0 == 1){
  b0 = !b0;
  b1 = !b1;
  int p = 1;
  f0 = a0 ^ b0 ^ p ;
  int cin = (a0 & b0)|(p & b0)|(a0 & p);
  f1 = cin ^ a1 ^ b1;
  c_out = (cin & a1)|(b1 & a1)|(cin & b1);
  digitalWrite(13,f0);
  digitalWrite(12,f1);
  digitalWrite(11,c_out);
}
```

```
  }
  else if(s2 == 1  && s1 == 1 && s0 == 0){
    int p = 1;
    int d0 = !a0;
    f0 = a0 ^ p ;
    int borrow = d0 & p;
    f1 = a1 ^ borrow;
    c_out = (!a1) & borrow;
    digitalWrite(13,f0);
    digitalWrite(12,f1);
    digitalWrite(11,c_out);

  }

}
```

THANK YOU