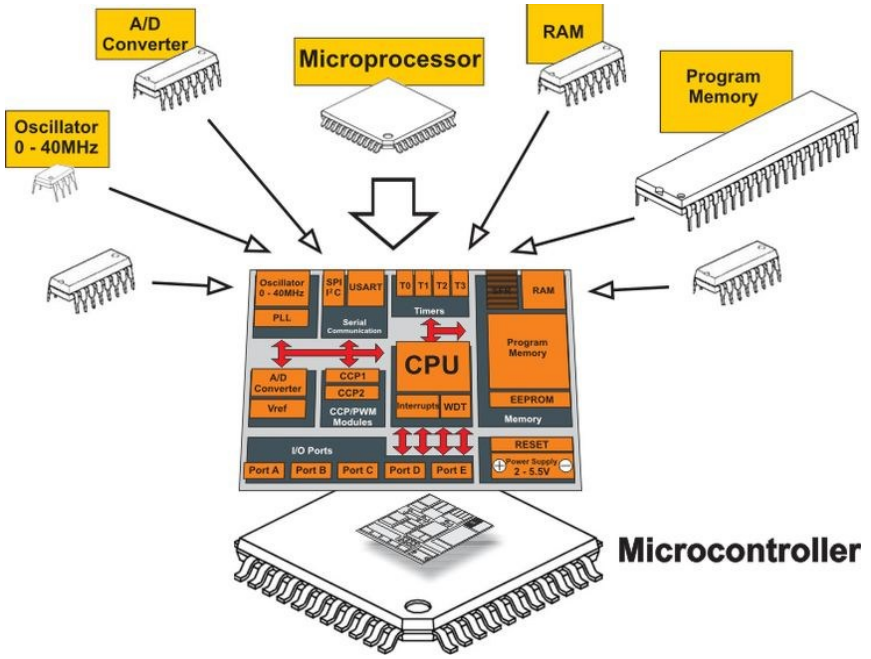
Robust Mechatronics

Microcontrollers

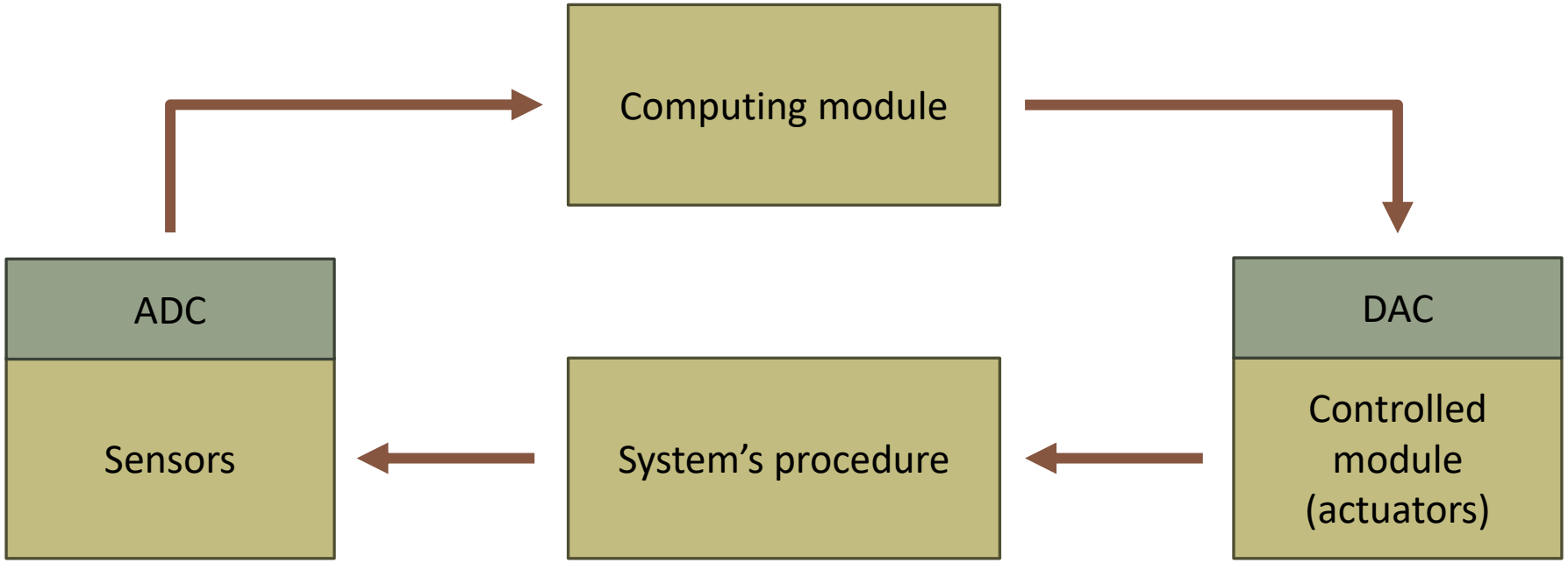


Dr Loukas Bampis, Assistant Professor

Mechatronics & Systems Automation Lab

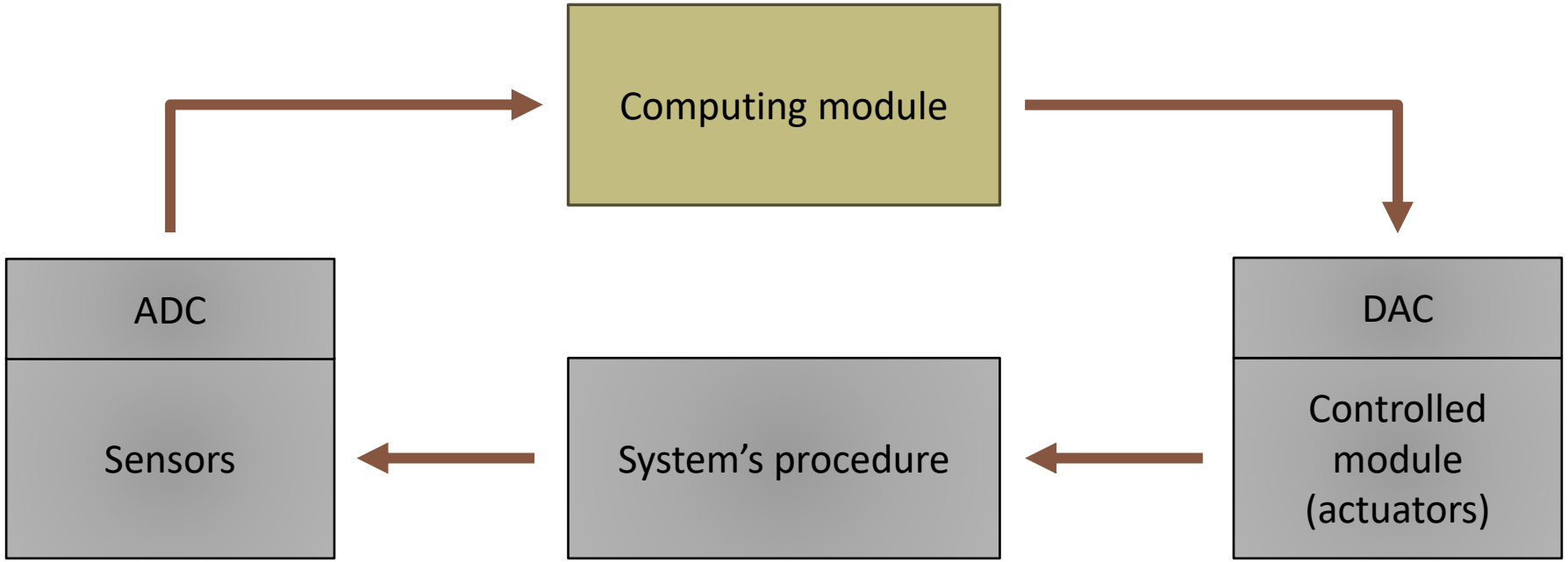
Microcontrollers

Control loop



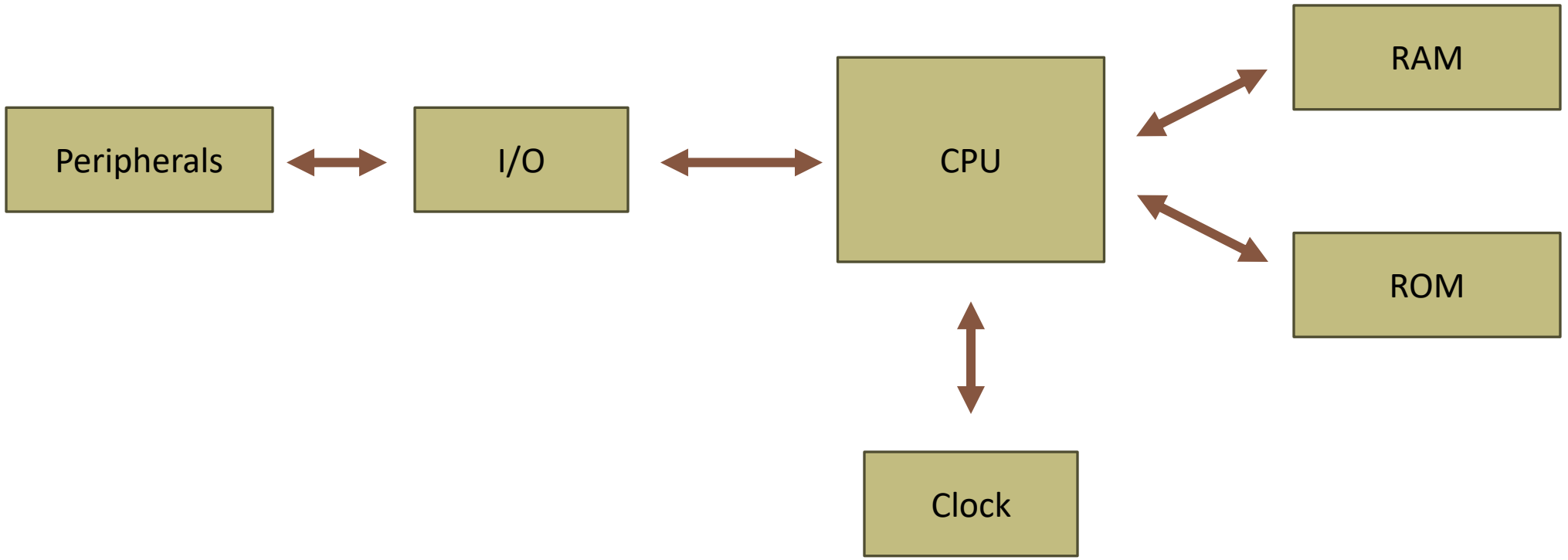
Microcontrollers

Control loop



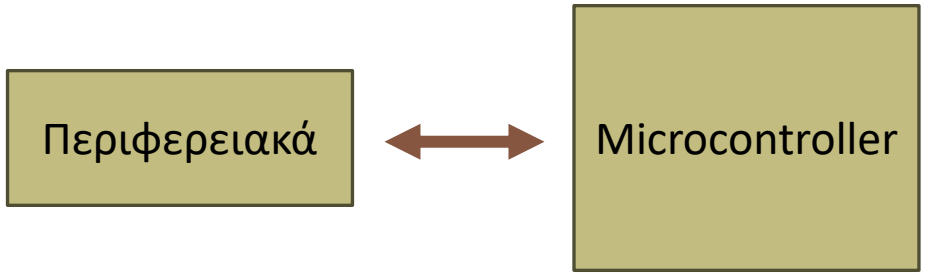
Microcontrollers

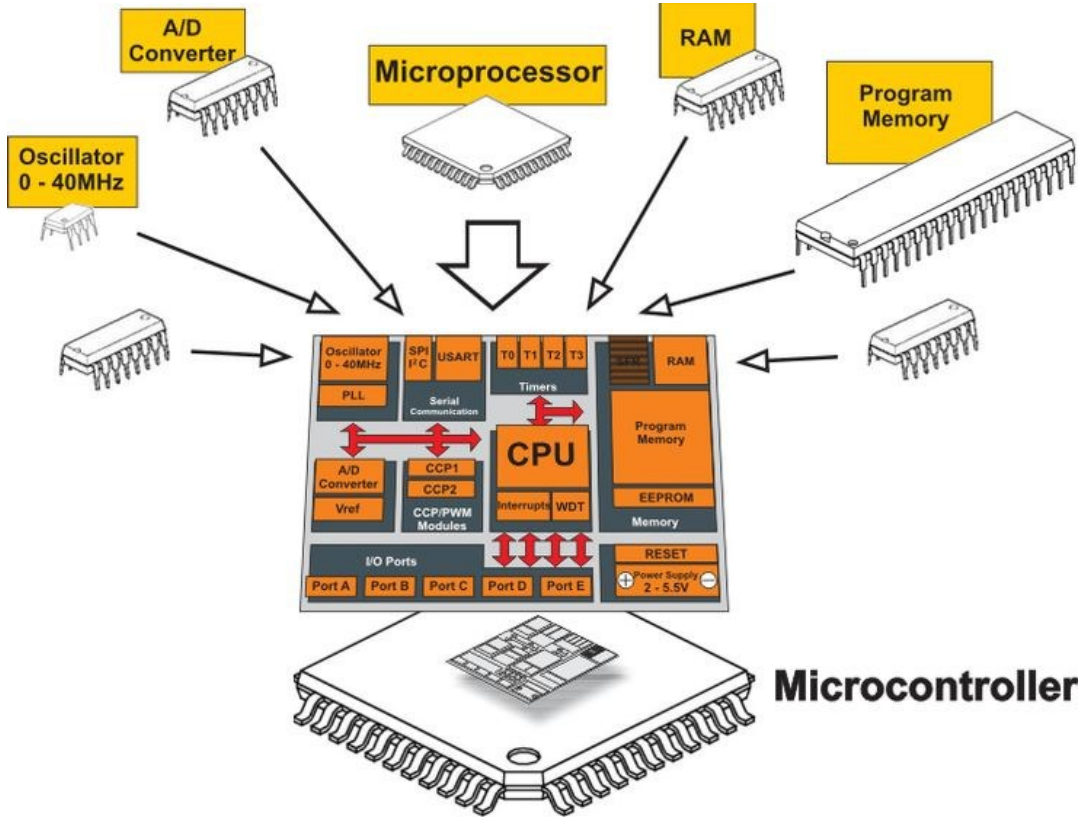
Microprocessor-based computing system



Microcontrollers

Microcontroller-based computing system





Microcontrollers

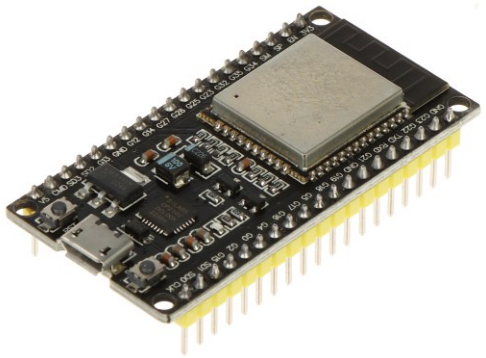
Characteristics

- Operating Frequency (MHz)
- Operating Voltage/Current (Volt & Ampere dc or AC)
- Data Bus (bits number)
- Address Bus (bits number)
- Memory (Bytes)
- ADC (**A**nalog to **D**igital **C**onverter)
- DAC (**D**igital to **A**nalog **C**onverter)
- PWM (**P**ulse **W**idth **M**odulation)
- Communication (wireless or wired)

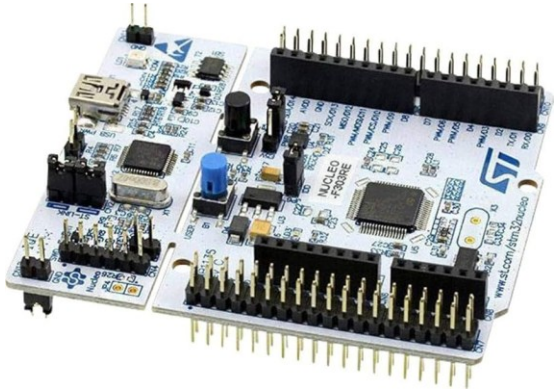
Microcontrollers

Examples of Microcontrollers

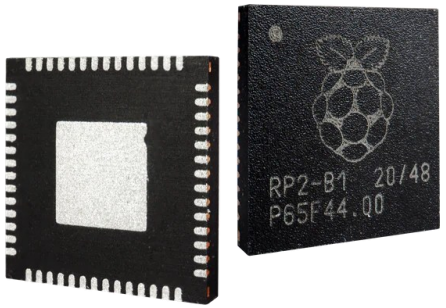
ESP32



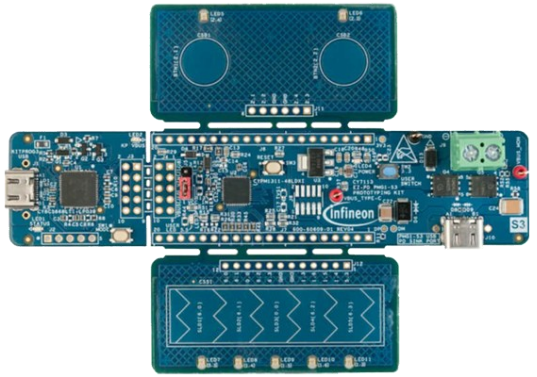
STM32



RP2040



Infineon EZ-PD™ PMG1-S3

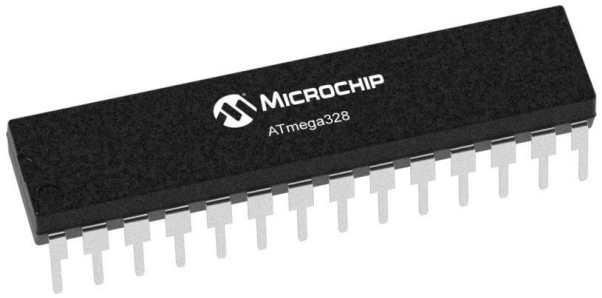


Microcontrollers

Arduino is an "open-source hardware/software" electronics prototyping platform, based on flexible and easy-to-use hardware and software intended for anyone who has:

- little programming experience
- elementary knowledge of electronics
- And is interested in creating interactive applications and systems.

~Creators of Arduino



Microcontrollers

Advantages:

Low cost:

- About 25 euros for the Arduino UNO
- Extra set of accessories (sensors, switches, monitors, cables) for less than ~40 euros
- Free Software

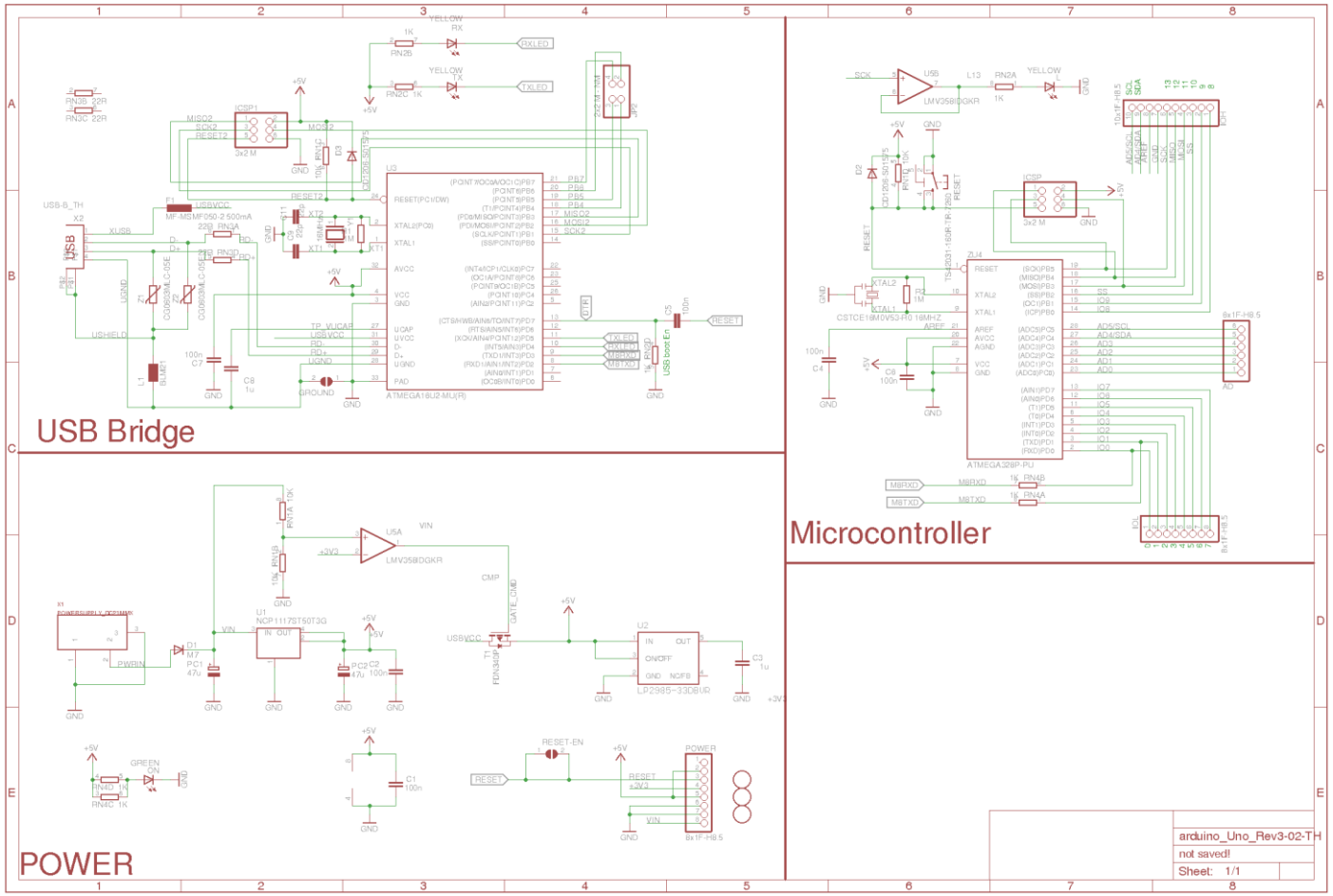
Compatibility: The software runs on Windows, OSX, and Linux

Low complexity: The development environment is simple to use by novice users

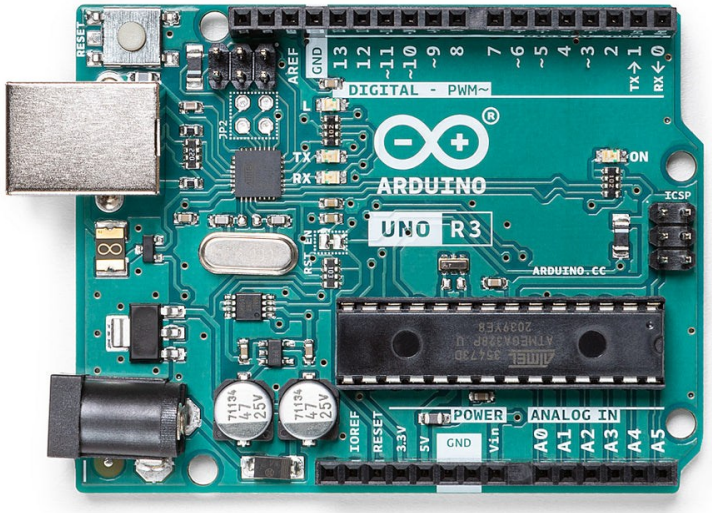
Extensible Open-Source Software: Codes and libraries are open-source which means that one can study and modify them according to their needs. New libraries may be added.

Microcontrollers

Open Hardware Platform



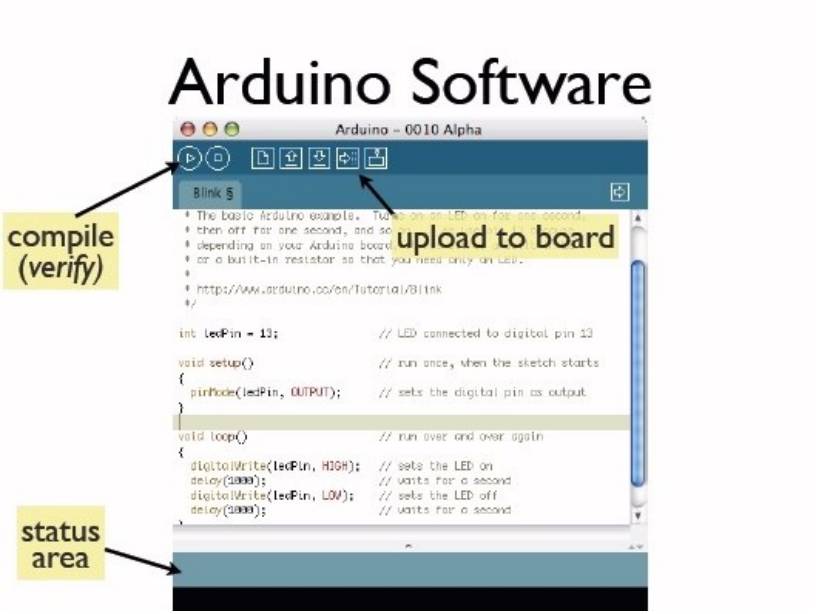
Microcontrollers



Hardware

Software

Integrated Development Environment - IDE



Microcontrollers

Arduino Characteristics

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

Arduino UNO R3
From the manufacturer's page

Microcontrollers

Arduino Characteristics

ATmega328: 8-bit microcontroller, with timing at 16MHz.

The ATmega328 has three types of built-in memory:

SRAM 2Kb

- It is used by programs to store variables, tables, etc. during runtime.
- As with a computer, this memory loses its data when the power supply to the Arduino stops or if it is reset.

EEPROM 1Kb (Electrically Erasable Programmable Read-Only Memory)

- It is used to write/read data from the user's programs during their runtime.
- Unlike SRAM, EEPROM does not lose its contents with a loss of power or reset. So it's the analogue of the hard drive.

Microcontrollers

Arduino Characteristics

Flash Memory 32Kb:

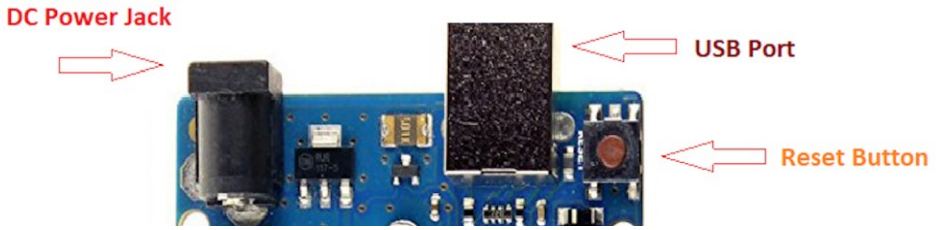
- 2Kb are used by the Arduino firmware installed by its manufacturer.
  - Necessary for the installation of the user's programs on the microcontroller via the USB port, i.e. without the need for an external hardware programmer.
  - In Arduino terminology it is called a bootloader.

- 30Kb are used to store these exact programs, after they have been compiled on your computer.

Flash memory, like EEPROM, does not lose its contents when power is lost or reset. Also, Flash memory is not normally intended for use during the runtime of our programs.

Microcontrollers

Arduino Characteristics



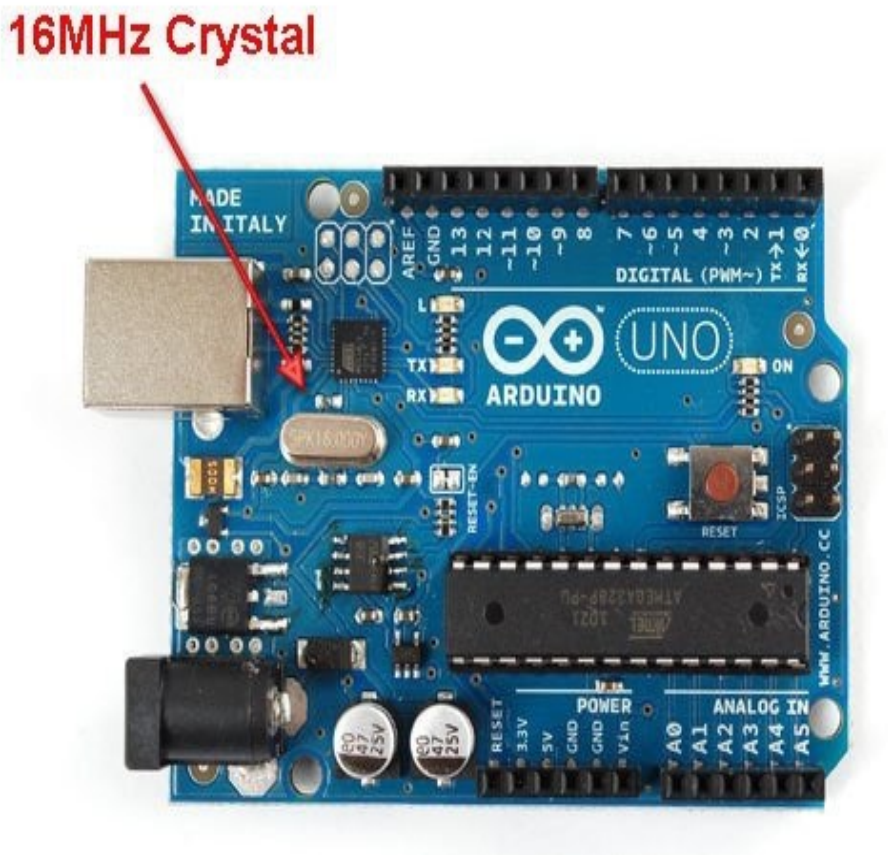
Power In

- By your PC, via USB connection
- Through external power supply supplied via a 2.1mm Barrel jack socket (positive pole in the center)
- <u>The external power supply should be from 7 to 12V.</u>
  (It can come from a typical commercial transformer, batteries, or any other DC source.)

Microcontrollers

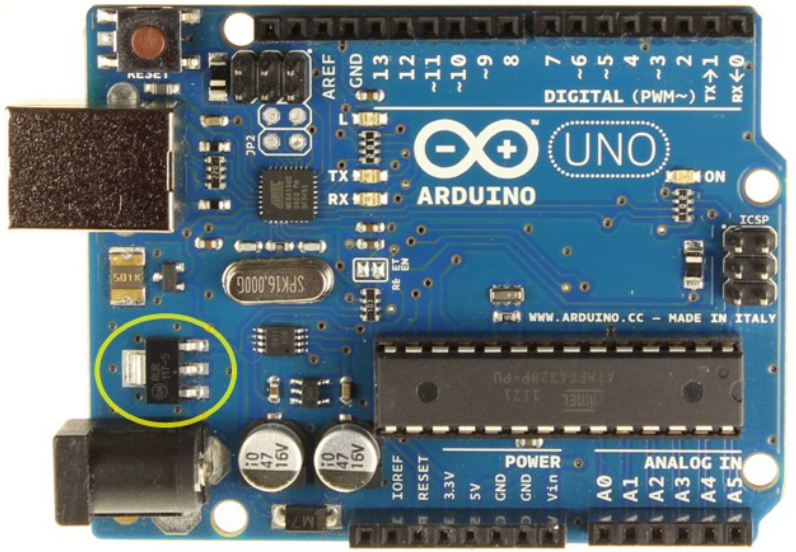
Arduino Characteristics

16MHz Crystalline Oscillator
To Provide Timing Pulses
(clock)



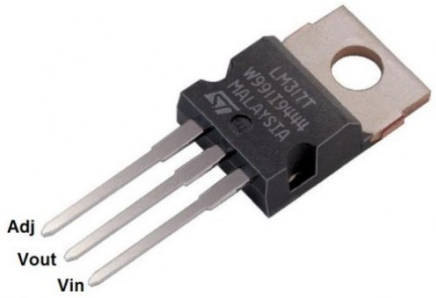
Microcontrollers

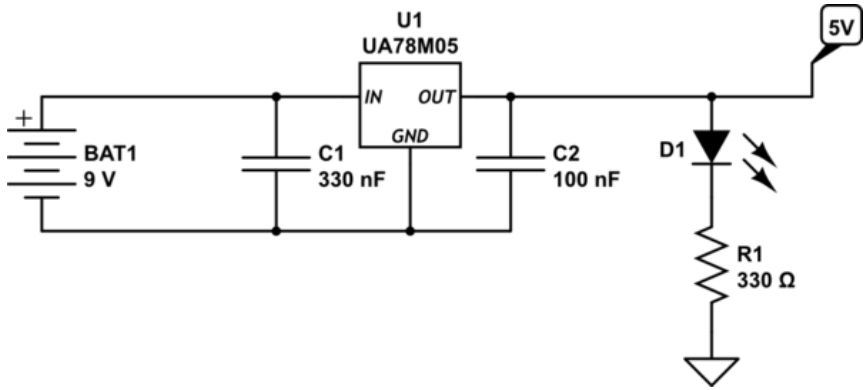
Arduino Characteristics



DC Linear Voltage Regulator (Stabilizer) 5V

- Input Voltage Range (Vin): 5.3V to 35V
- Output Voltage Range (Vout): 3.3V to 12V
- Output Current (Iout): έως 500mA
- With built-in current limiting circuit – Short circuit and overheat protection

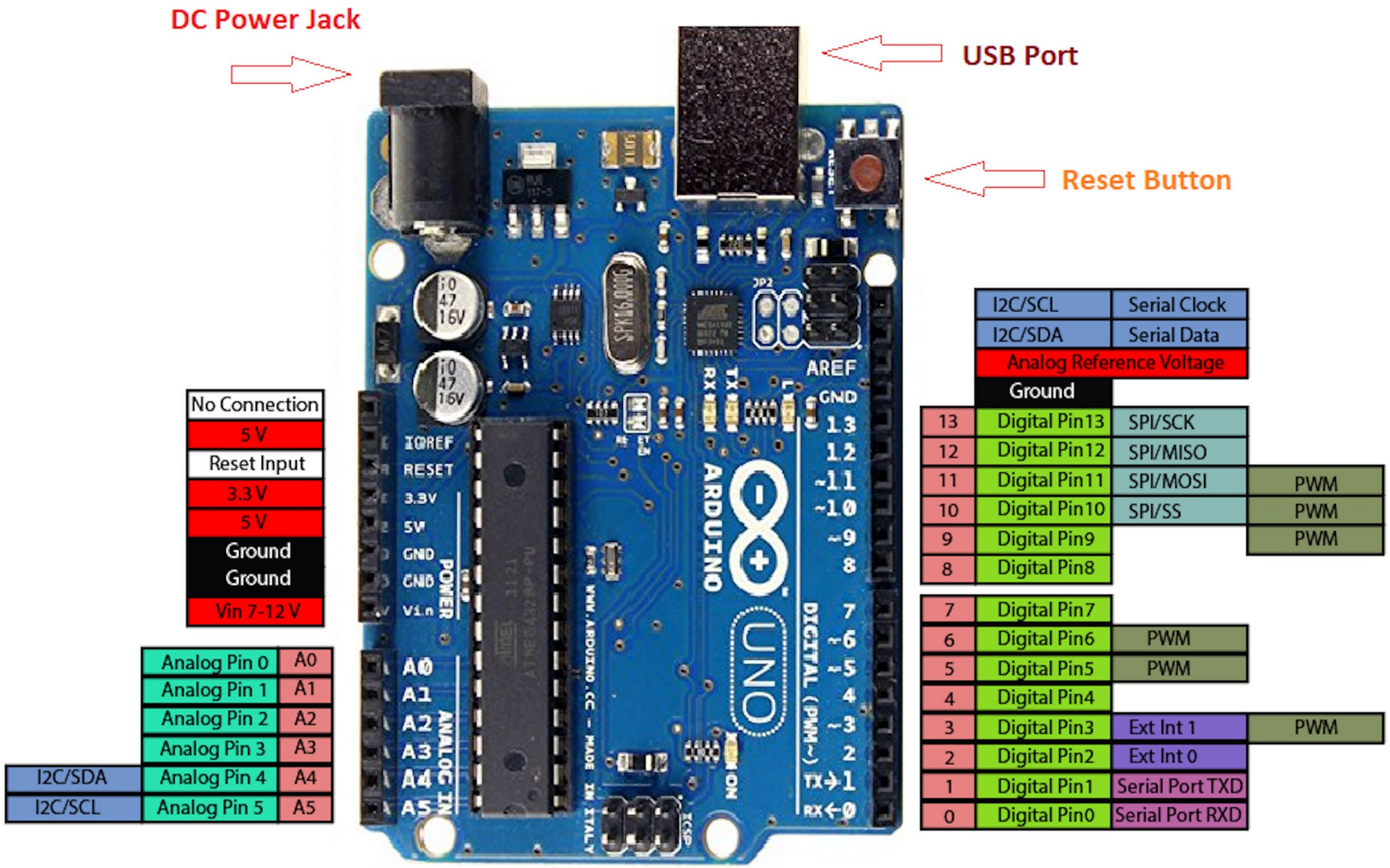




Microcontrollers

Arduino Characteristics

Inputs – Outputs



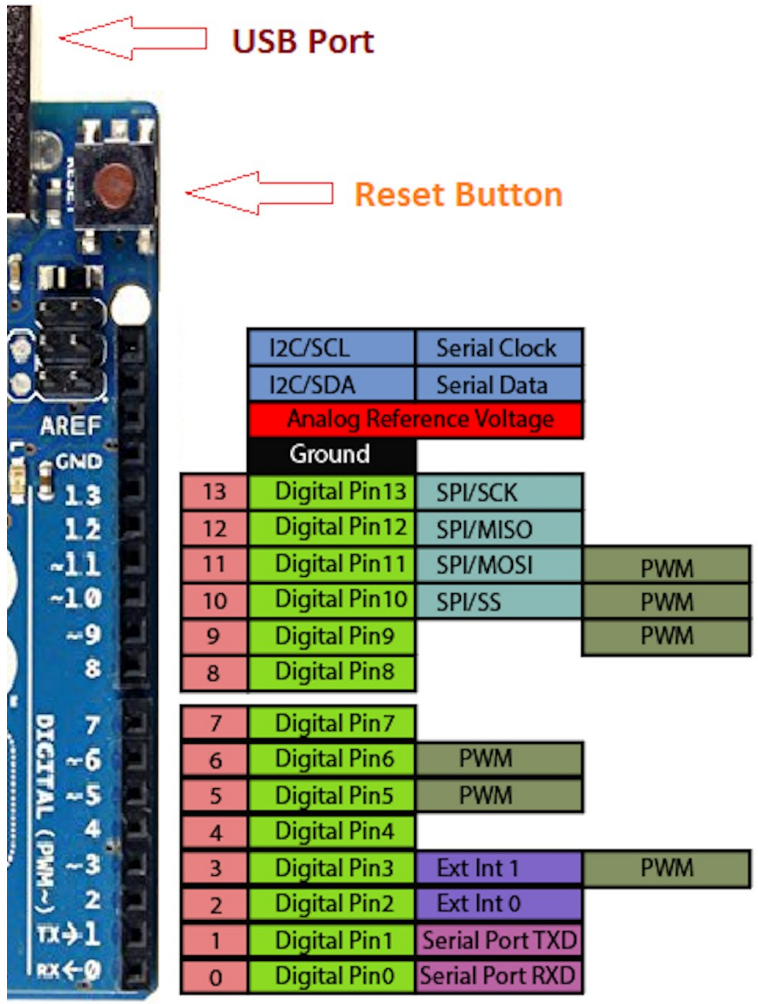
Microcontrollers

Arduino Characteristics

Digital Inputs/Outputs - BASIC FUNCTIONALITY

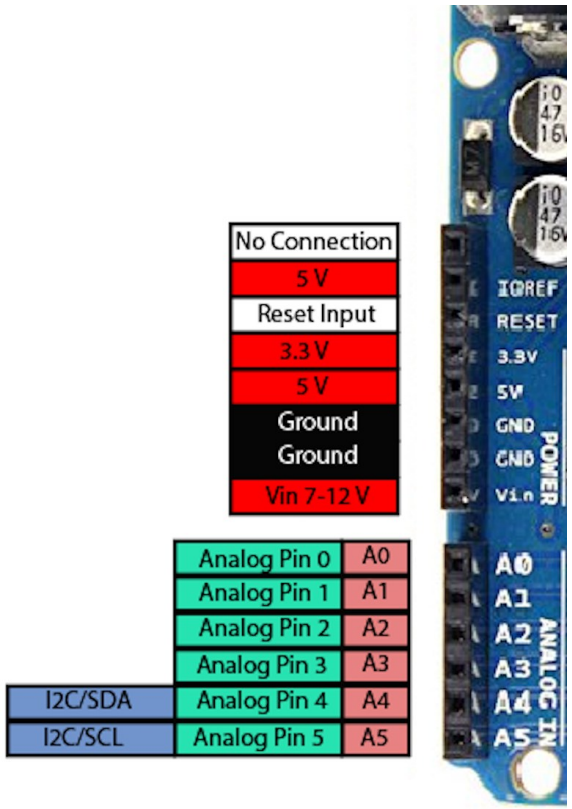
14 Pins (female) 0 ~ 13

- Function either as inputs or outputs
- Operation at 5 Volt DC, up to 40 mA
- 2 Modes: High & Low (0 & 5V, respectively)
- Function setting (input/output) through the algorithm



Microcontrollers

Arduino Characteristics



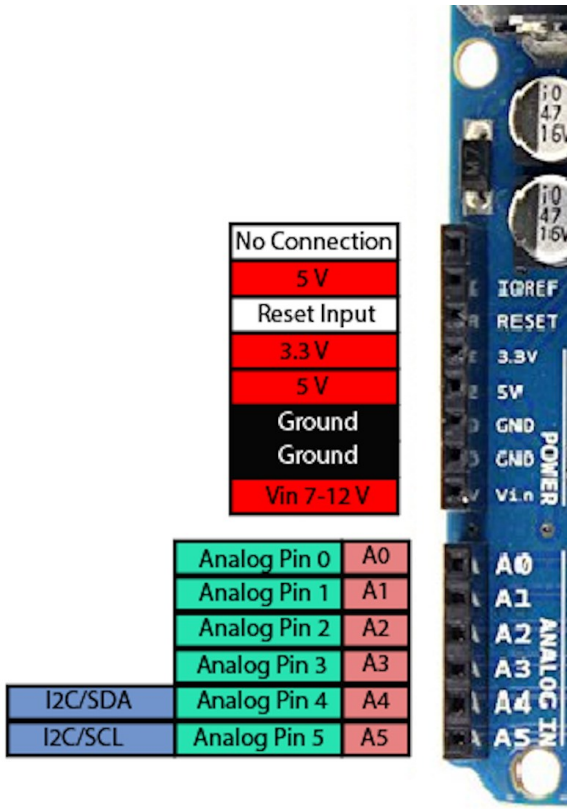
Analogue Inputs - BASIC FUNCTIONALITY

6 Pins (female) 0 ~ 5

- Function as analog inputs
- Function as digital inputs/outputs
- Programming of their operation (analogue/digital) through the algorithm
  (In this case the pins are renamed from 0~5 to 14~19 respectively).

Microcontrollers

Arduino Characteristics



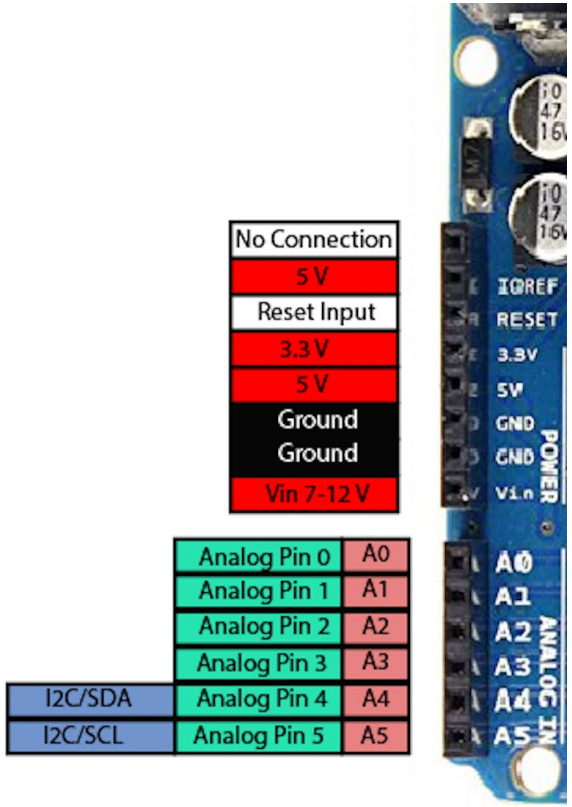
Analogue Inputs - BASIC FUNCTIONALITY

Analog to Digital Converter – ADC

- Number of bits: 10
- Reference Voltage: 5V
- Number of levels: 1024
- Digital encoding range: 0 έως 1023
- Digital encoding: voltage input (V) * $\frac{1023}{5}$

Microcontrollers

Arduino Characteristics



POWER PINs

- The RESET indicator, when grounded (on any of the 3 pins with the GND indicator present on the Arduino) results in the restart of the Arduino.
- The 3.3V indicator, can supply the respective components with 3.3V.
- 5V indicator, can power the additional components with 5V.
- The GND indicator refers to the ground (reference voltage).
- The Vin indicator has a double role:
  - Combined with the ground pin next to it, it can work as an external power method of the Arduino
  - If there is already an external power supply, it can be used to power components with the full voltage of the external power supply (7~12V) (before it passes through the voltage regulator as is done with the 5V pin)

Microcontrollers

Arduino Characteristics

Two functions must be defined to make a circular execution program:

- setup(): a function executed once at the beginning of the program that initializes settings and variables
- loop(): a function that is executed repeatedly until the microcontroller is disabled or given a proper shutdown command by the algorithm

```
#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04

// defines variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
  Serial.println("with Arduino UNO R3");
}
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);  // HIGH pulse with duration (in microseconds) equal to the time
                                      // from the sending to the reception of the signal.

  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay (1000) ;
}
```

Microcontrollers

Remember…

Variable Type	Type of information stored (Examples)	Default values	Notes
Boolean	Μία τιμή αληθείας (1, TRUE, HIGH) ή ψεύδους (0, FALSE, LOW)	0, FALSE, LOW	Truth values are written in English in capital letters
int	Έναν ακέραιο αριθμό (-5, 15, 1047, etc.)	0	Positive or negative value
double	Έναν δεκαδικό αριθμό (-0.5, 123.77, etc.)	0	Positive or negative value
float	Έναν δεκαδικό αριθμό (-0.5, 123.77, etc.)	0	Positive or negative value
char	Έναν χαρακτήρα (‘c’, ‘A’, ‘5’, ‘?’, etc.)	Unspecified	They must be written in single quotation marks
string	Μία σειρά χαρακτήρων (“Hello World!”, “10”, “157+5”, etc.)	Empty (“”)	They must be written in double quotation marks

Microcontrollers

Remember…

Operator	Functionality	Notes
=	Assign a value to a variable	Basic arithmetic operations
+	Adding Two or More Values	
-	Subtracting two or more values	
*	Multiplying Two or More Values	
/	Dividing Two or More Values	
++	Increase by 1	Commonly used in iteration loops
--	Decrease by 1	
==	Check if two values are equal	Usually used in Boolean conditions
!=	Check if two values are different	
> or <	Compare if one value is less/higher than another	
<= or >=	Compare whether one value is less/greater than or equal to another	
&& or ||	Logical AND/Logical OR between successive Boolean conditions	

Microcontrollers

Remember…

Variables Definition

```
1  // Example of constants
2  #define secondLedPin 7
3  const int firstLedPin = 7;
4  const string greetingMessage = "Automation is Fun!";
5
6  // Example of variables
7  int sensorValue = 0;
8  char myStringVarialbe = "Hello there!";
```

Condition Statements

```
17    if(sensorValue == 0){
18      // do something
19    }else if(sensorValue == 1){
20      // do something else
21    }else
22      // do something in one line
```

Iterative Procedures

```
17    while(sensorValue == 0){
18      delay(1000);
19    }
20    for(int i = 0; i<100; i++){
21      Serial.println(i);
22    }
23    for(int i = 99; i>=0; i--){
24      Serial.println(i);
25    }
```

Function Definition

```
15  int addOne(int num){
16    int newNum = num + 1;
17    return newNum;
18  }
19  void loop()
20  {
21    int printNum = addOne(4);
22    Serial.print(printNum);
23  }
```

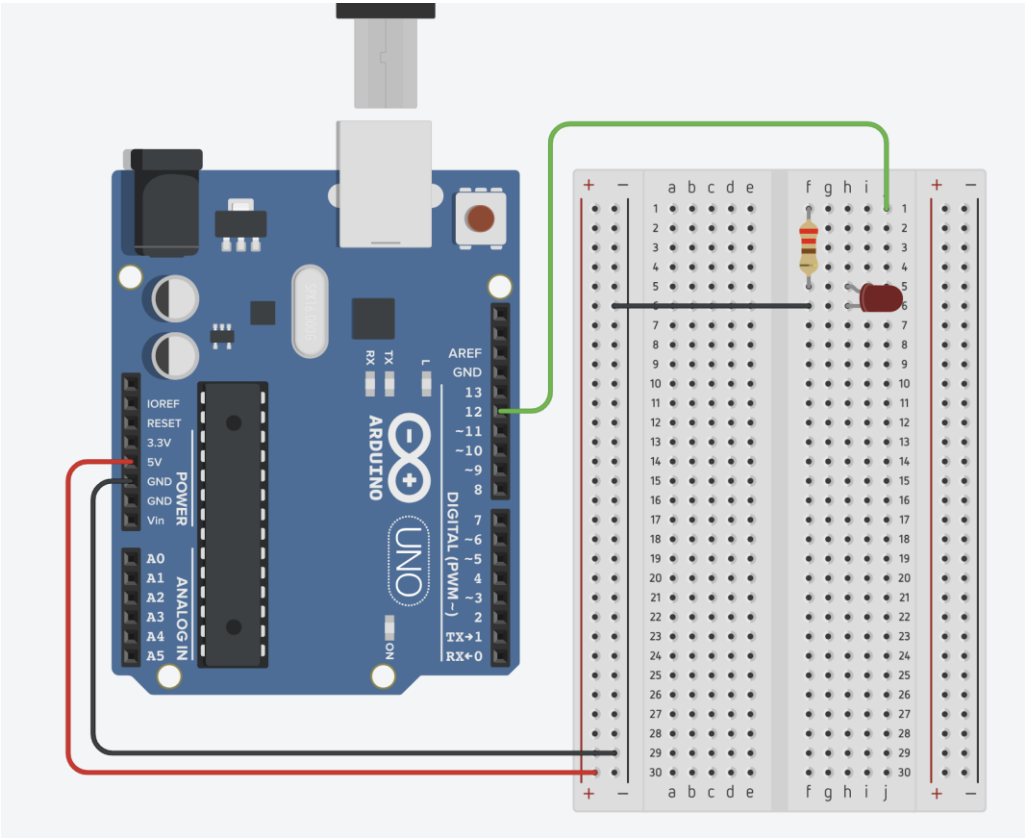
Microcontrollers

Basic functions offered by Arduino

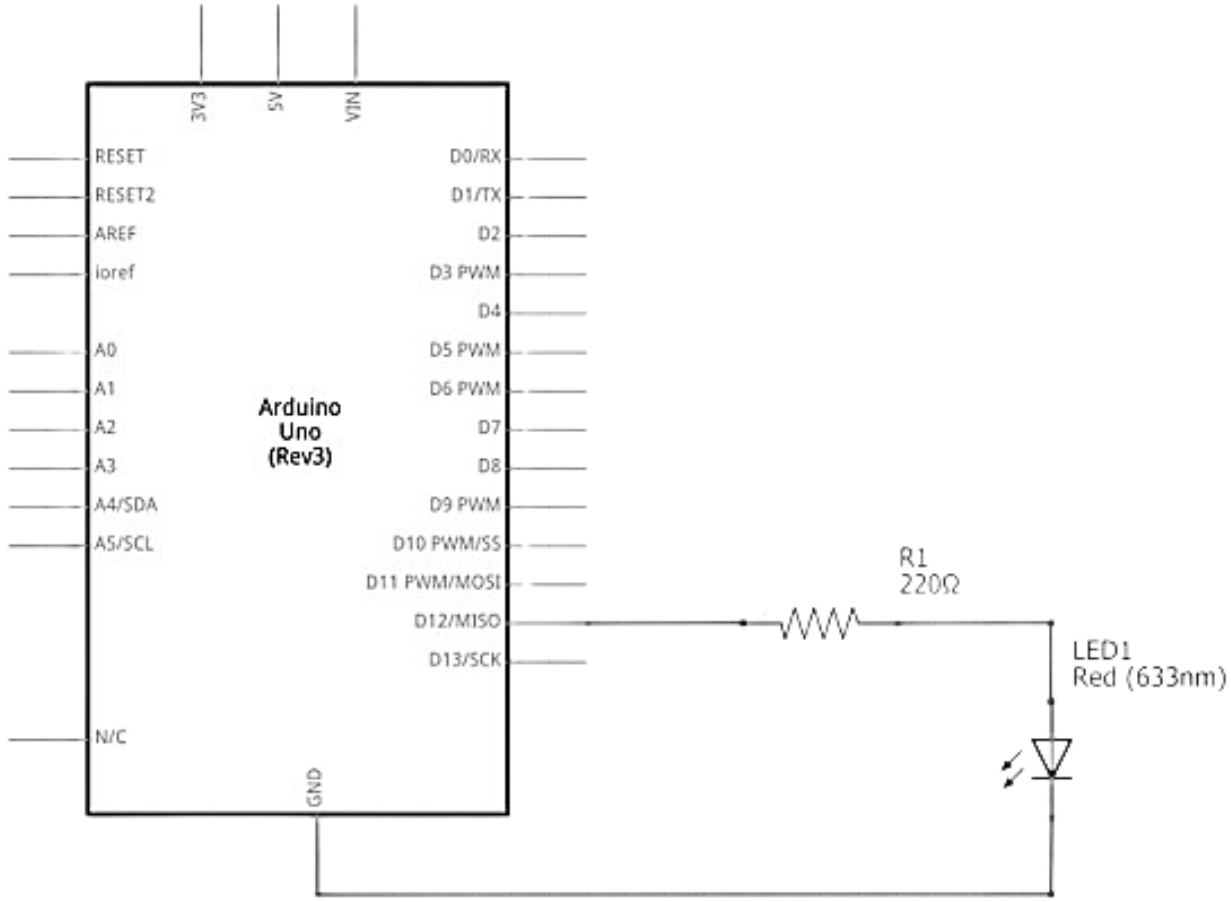
Function	Explanation
pinMode(pin,mode)	Set a pin as INPUT or OUTPUT (mode)
digitalWrite(pin, value)	Set a Digital Output Pin to HIGH or LOW (value)
value = digitalRead(pin)	Read the value of a digital input PIN as HIGH or LOW (value)
analogWrite(pin, value)	Set an analog output pin to 0-255 (value)
value = analogRead(pin)	Read the value of an analog input pin as a value of 0-1023 (value)
delay(milliseconds)	Pause the program for a certain amount of time (ms)
Serial.begin(value)	Initialize the Serial Port at a rate equal to value
Serial.print(value)	Printing the value variable to the serial port
Serial.println(value)	Printing the value variable to the serial port and change line

Microcontrollers

Schematic Circuit Diagram (TinkerCAD)



Circuit Diagram

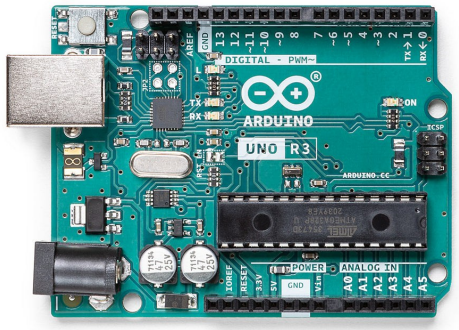


Microcontrollers

Getting Started with Arduino

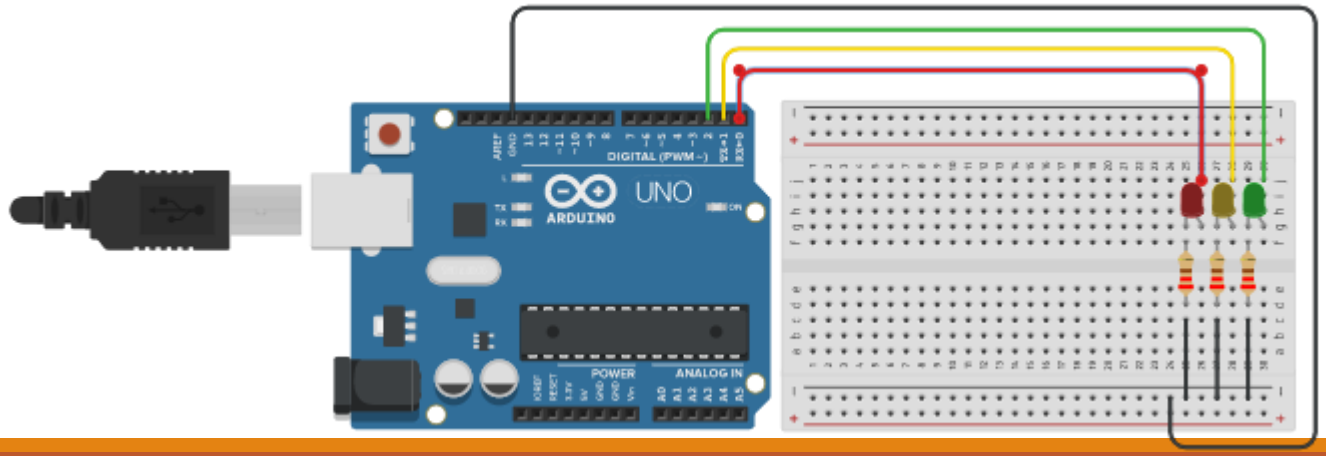
Given the hardware

- Installing theArduino IDE
  https://www.arduino.cc/en/software
- Connecting a board via USB Type B cable



In a simulation environment

- Autodesk TinkerCAD
  https://www.arduino.cc/en/software
- Microsoft Maker Code
  https://maker.makecode.com
- WOKWI
  https://wokwi.com

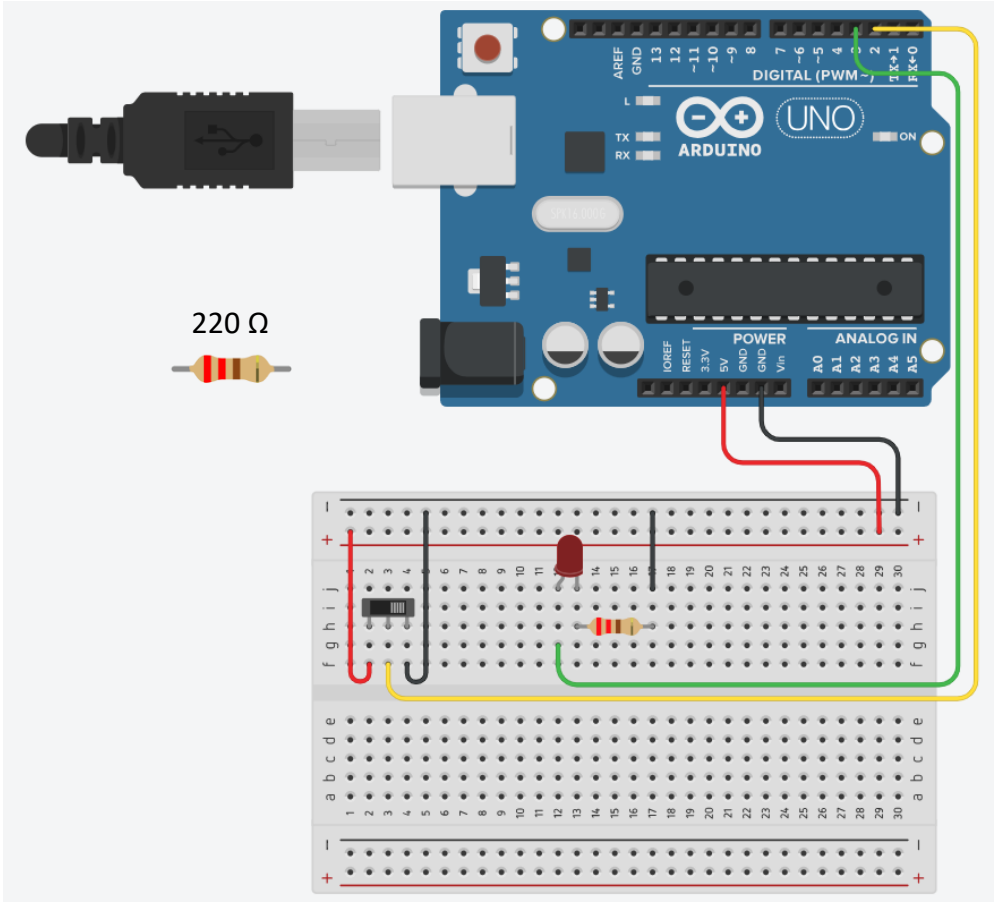


Microcontrollers

Examples

```
1   const int Switch_Pin = 2;
2   const int LED_Pin = 3;
3   void setup()
4   {
5     pinMode(Switch_Pin, INPUT);
6     pinMode(LED_Pin, OUTPUT);
7     Serial.begin(9600);
8     Serial.println("Simple Logic Circuit");
9   }
10  void loop()
11  {
12    int Switch = digitalRead(Switch_Pin);
13    Serial.println(Switch);
14
15    if(Switch == 1)
16      digitalWrite(LED_Pin, HIGH);
17    else
18     digitalWrite(LED_Pin, LOW);
19
20    delay(50);
21  }
```

Simple Logic Circuit

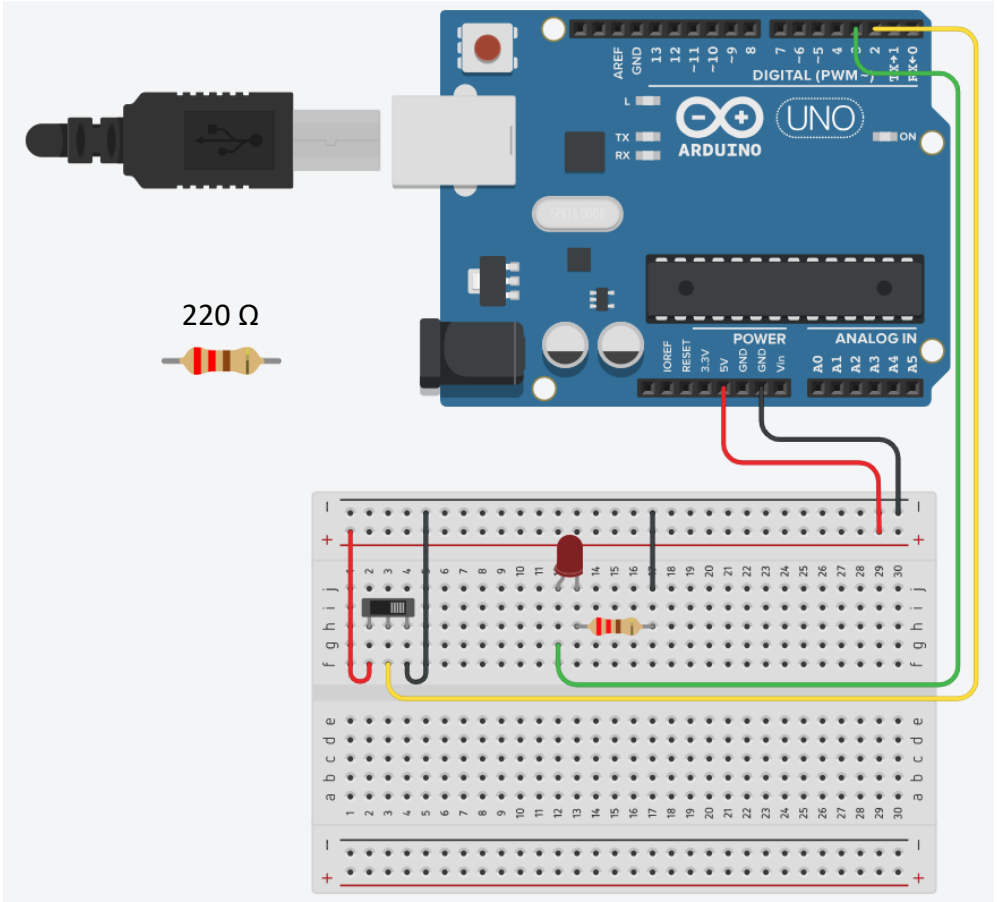


Microcontrollers

Examples

```
1   const int Switch_Pin = 2;
2   const int LED_Pin = 3;
3   void setup()
4   {
5     pinMode(Switch_Pin, INPUT);
6     pinMode(LED_Pin, OUTPUT);
7     Serial.begin(9600);
8     Serial.println("Simple Logic Circuit");
9   }
10  void loop()
11  {
12    int Switch = digitalRead(Switch_Pin);
13    Serial.println(Switch);
14
15    digitalWrite(LED_Pin, Switch);
16
17    delay(50);
18  }
```

Simple Logic Circuit (Ver 2)



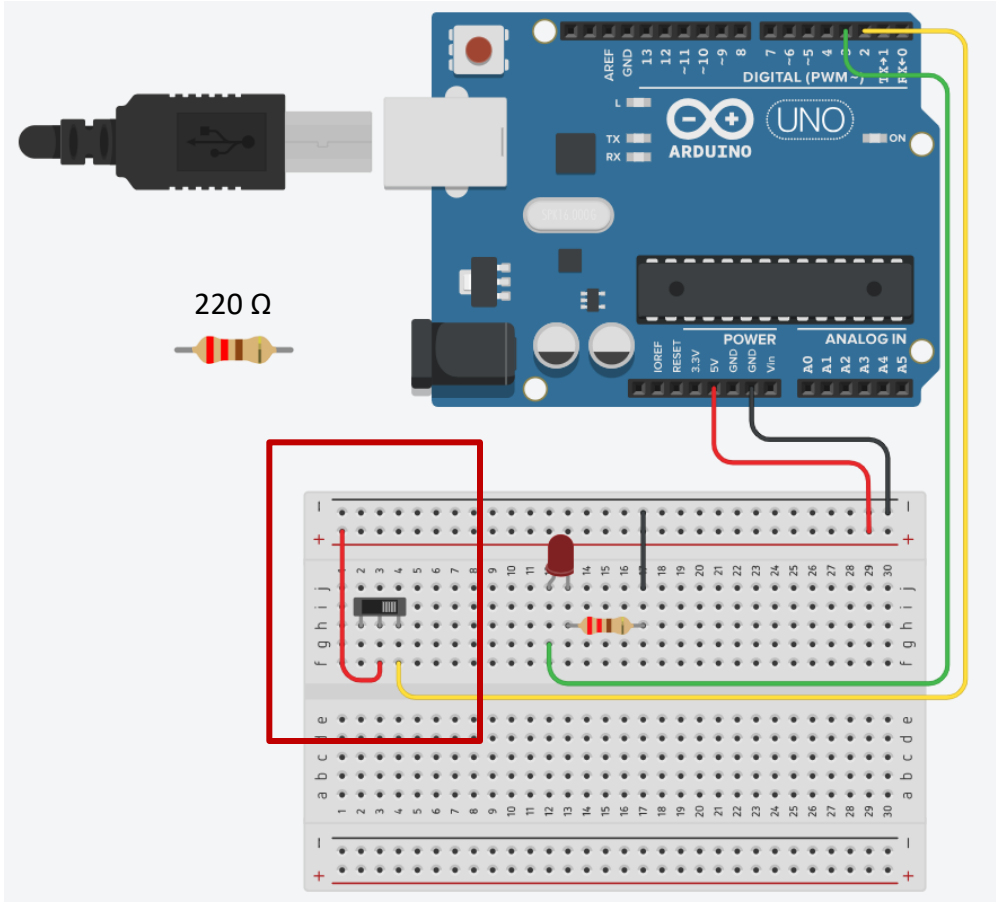
Microcontrollers

Examples

```
1   const int Switch_Pin = 2;
2   const int LED_Pin = 3;
3   void setup()
4   {
5     pinMode(Switch_Pin, INPUT);
6     pinMode(LED_Pin, OUTPUT);
7     Serial.begin(9600);
8     Serial.println("Simple Logic Circuit");
9   }
10  void loop()
11  {
12    int Switch = digitalRead(Switch_Pin);
13    Serial.println(Switch);
14
15    digitalWrite(LED_Pin, Switch);
16
17    delay(50);
18  }
```

Each pin must be connected either to V_{ref} or to GND

Simple Logic Circuit:: Common Errors



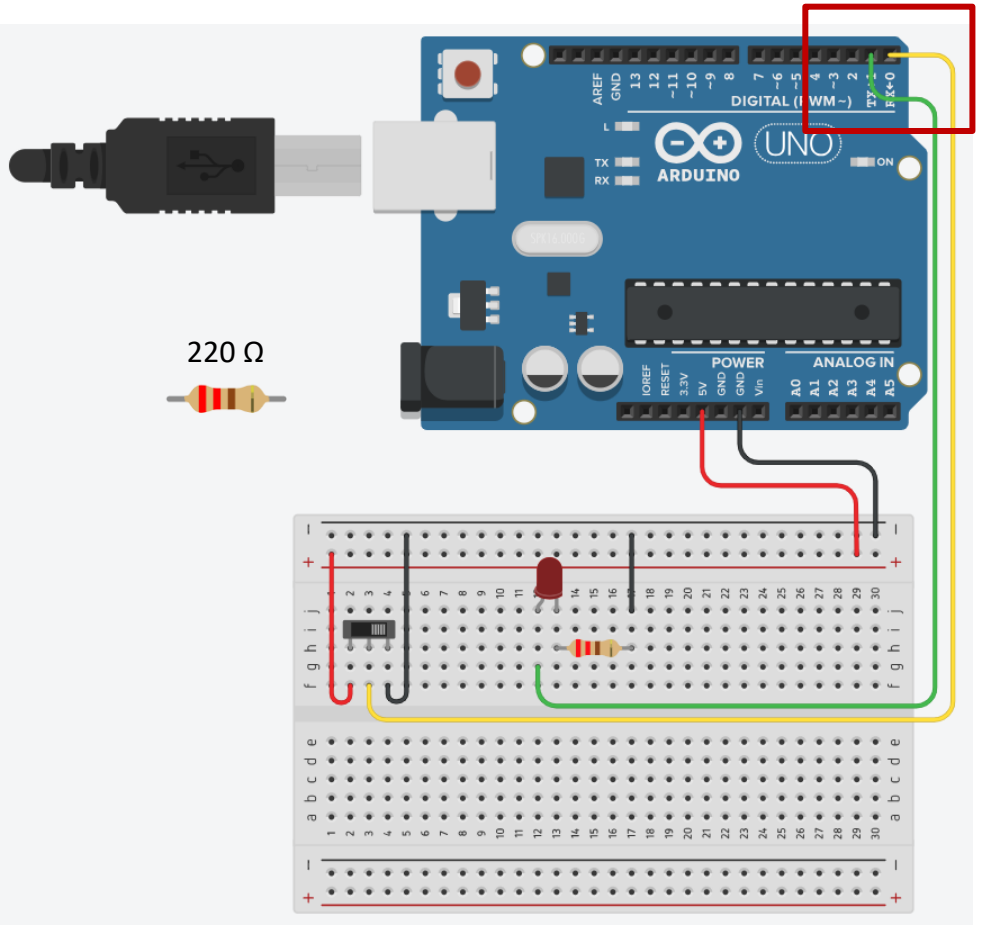
Microcontrollers

Examples

```
1   const int Switch_Pin = 2;
2   const int LED_Pin = 3;
3   void setup()
4   {
5     pinMode(Switch_Pin, INPUT);
6     pinMode(LED_Pin, OUTPUT);
7     Serial.begin(9600);
8     Serial.println("Simple Logic Circuit");
9   }
10  void loop()
11  {
12    int Switch = digitalRead(Switch_Pin);
13    Serial.println(Switch);
14
15    digitalWrite(LED_Pin, Switch);
16
17    delay(50);
18  }
```

Digital Pins 0 and 1 are also connected to the Serial Port. They should not be utilized if Serial.print() is used.

Simple Logic Circuit:: Common Errors



Microcontrollers

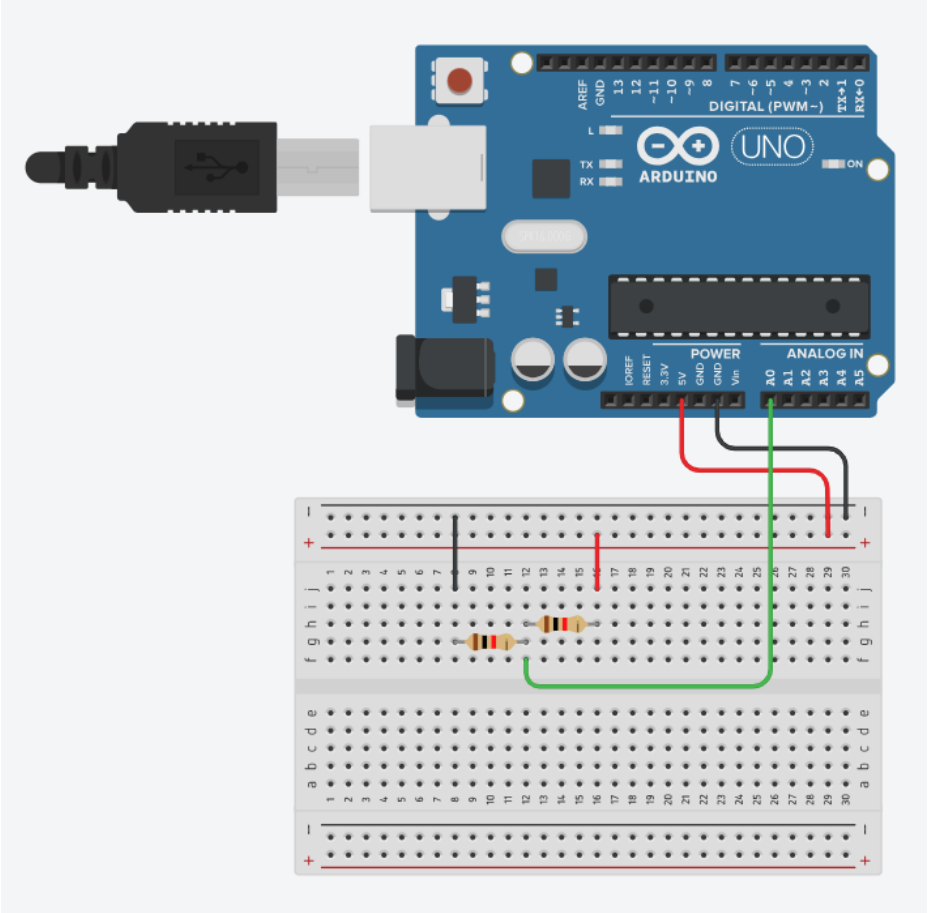
Examples

Simple Voltage Divider

```
1   int pot_pin = A0;
2   void setup()
3   {
4     Serial.begin(9600);
5     Serial.println("Simple Voltage Divider");
6   }
7   void loop()
8   {
9     int output = analogRead(pot_pin);
10    float voltage = (output/1024.0)*5.0;
11
12    Serial.print(voltage);
13    Serial.println(" V");
14  }
```

Analog to Digital Converter for Arduino: 10bit

$$V_{out} = \frac{b}{2^n} * V_{ref}$$



Microcontrollers

Examples

HC-SR04 Ultrasonic Distance Sensor

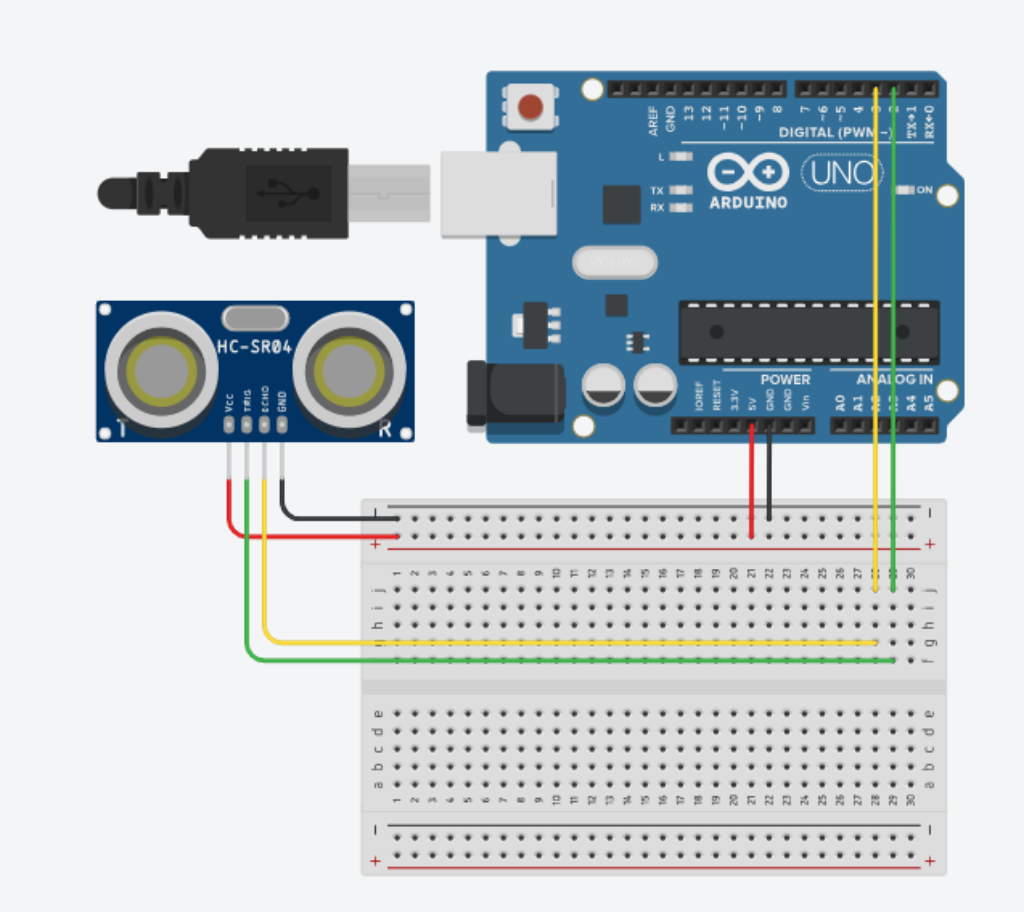
```
#define echoPin 3 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 2 //attach pin D3 Arduino to pin Trig of HC-SR04

// defines variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
  Serial.println("with Arduino UNO R3");
}
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);  // HIGH pulse with duration (in microseconds) equal to the time
                                      // from the sending to the reception of the signal.

  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay (1000) ;
}
```



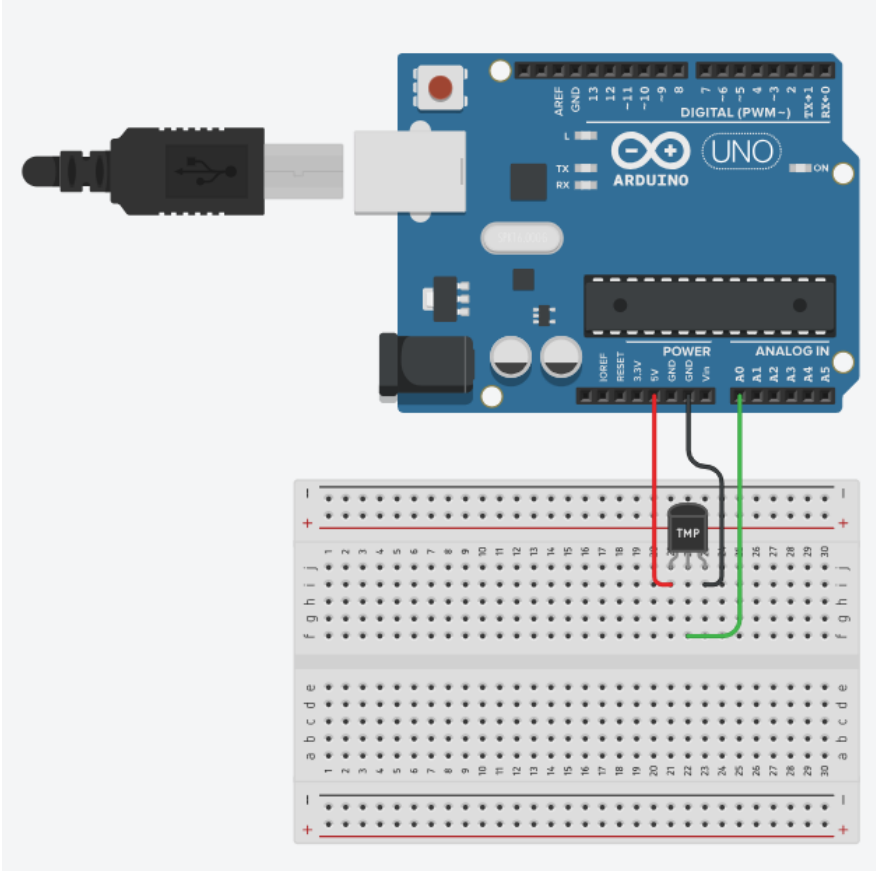
Microcontrollers

Examples

TMP36
Temperature Sensor

```
#define sensorPin A0
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
// the loop routine runs over and over again forever:
void loop() {
  // Get a reading from the temperature sensor:
  int sensorInput = analogRead(A0); //read the analog sensor and store it
  float voltage = sensorInput * (5.0 / 1024.0);
  float temperatureC = (voltage - 0.5) * 100;

  // print out the value you read:
  Serial.print("Temperature:");
  Serial.println(temperatureC);
  delay(1000);
}
```



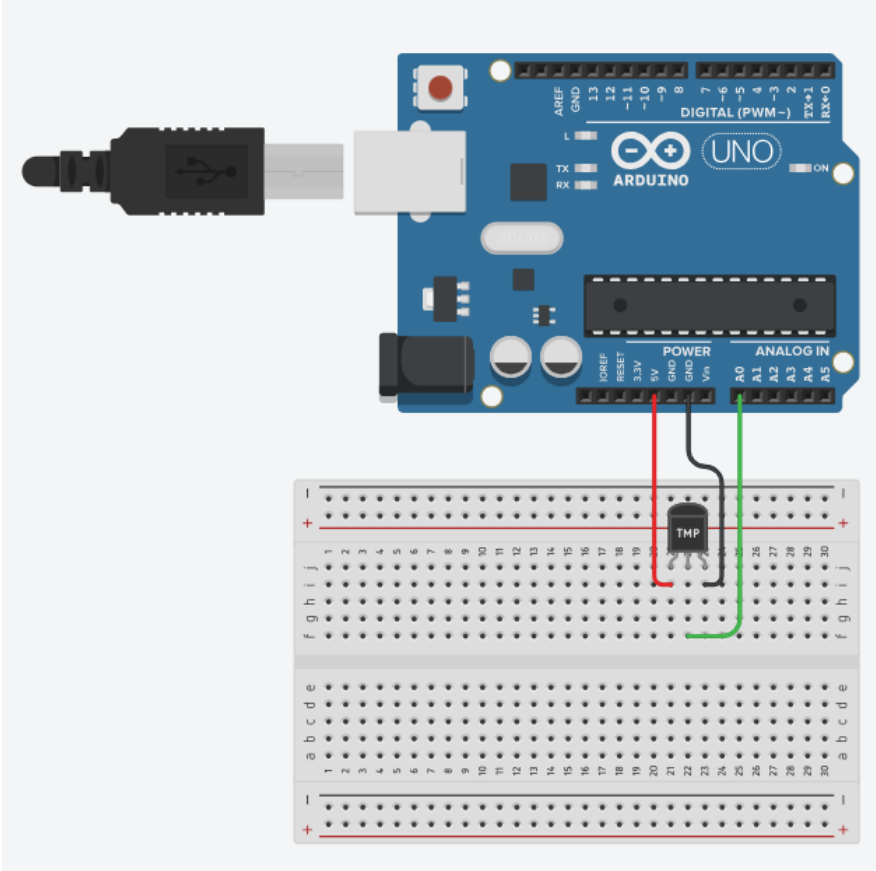
Microcontrollers

Examples

TMP36
Temperature Sensor

```
#define sensorPin A0
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
// the loop routine runs over and over again forever:
void loop() {
  // Get a reading from the temperature sensor:
  int sensorInput = analogRead(A0); //read the analog sensor and store it
  float voltage = sensorInput * (5.0 / 1024.0);
  float temperatureC = (voltage - 0.5) * 100;

  // print out the value you read:
  Serial.print("Temperature:");
  Serial.println(temperatureC);
  delay(1000);
}
```



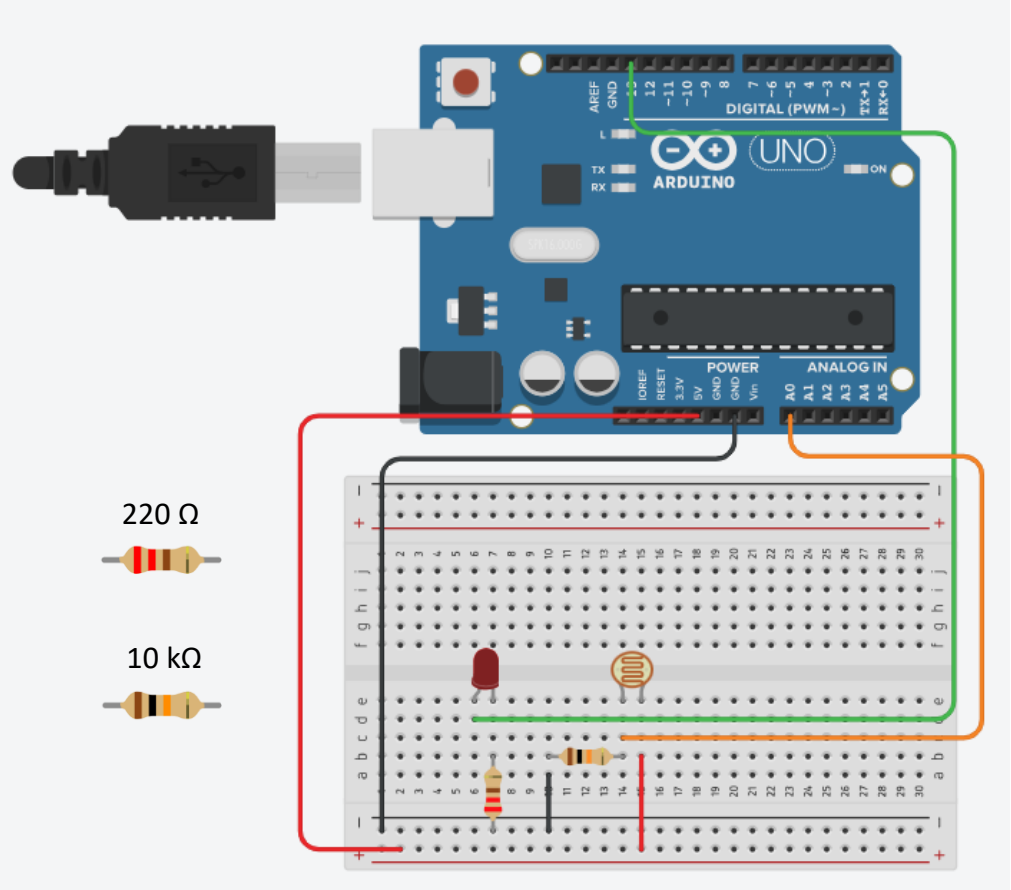
Which is the Transfer Function for TMP36?

Microcontrollers

Examples

LDR - Light-Dependent Resistor
Light Intensity Sensor

```
1   const int ledPin = 13;
2   const int ldrPin = A0;
3   void setup() {
4     Serial.begin(9600);
5     pinMode(ledPin, OUTPUT);
6     pinMode(ldrPin, INPUT);
7   }
8   void loop() {
9     int ldrStatus = analogRead(ldrPin);
10
11    if (ldrStatus <= 900)
12    {
13      digitalWrite(ledPin, HIGH);
14      Serial.print("Its Dark, Turn on the LED:");
15      Serial.println(ldrStatus);
16
17    }
18    else
19    {
20      digitalWrite(ledPin, LOW);
21      Serial.print("Its Bright, Turn off the LED:");
22      Serial.println(ldrStatus);
23    }
24  }
```



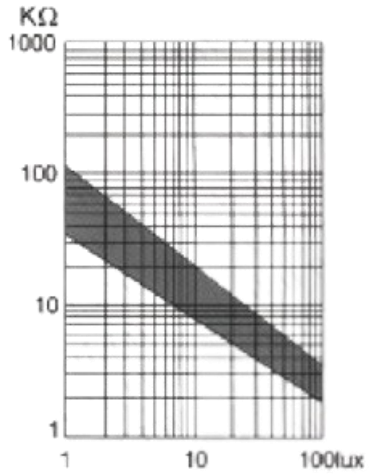
Microcontrollers

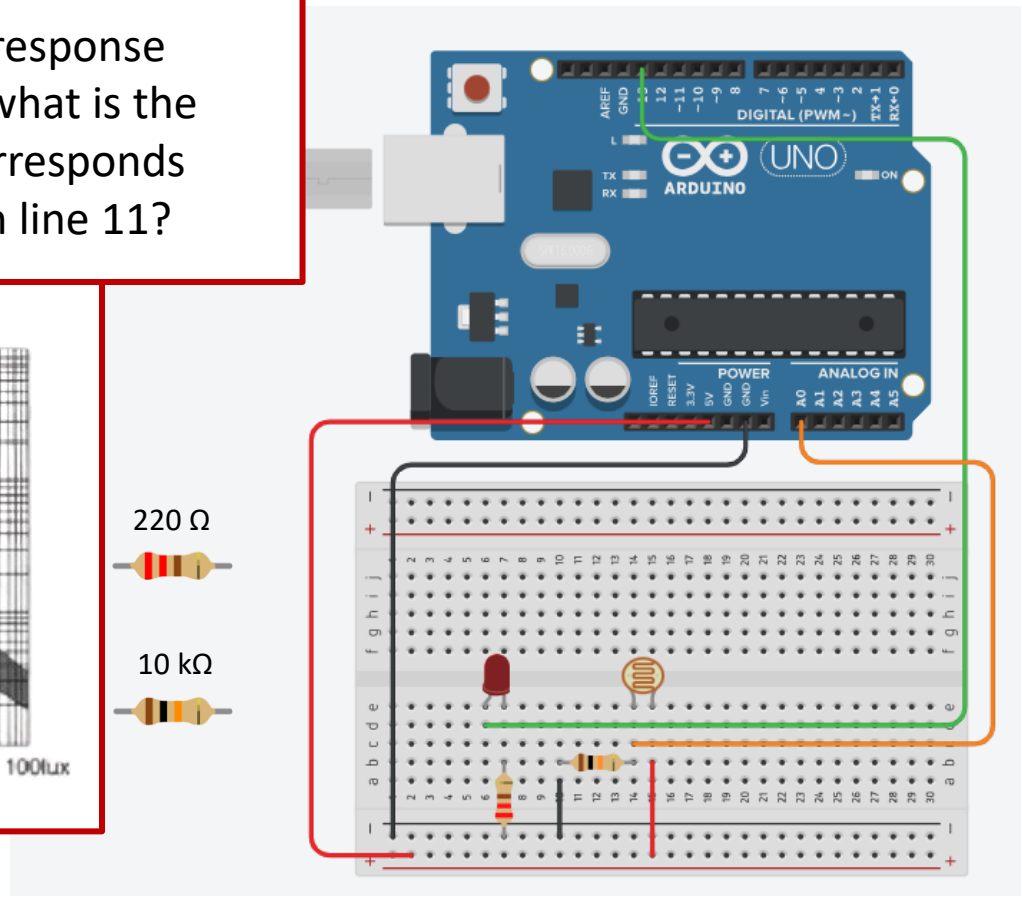
Examples

LDR - Light-Dependent Resistor
Light Intensity Sensor

```
1   const int ledPin = 13;
2   const int ldrPin = A0;
3   void setup() {
4     Serial.begin(9600);
5     pinMode(ledPin, OUTPUT);
6     pinMode(ldrPin, INPUT);
7   }
8   void loop() {
9     int ldrStatus = analogRead(ldrPin);
10
11    if (ldrStatus <= 900)
12    {
13      digitalWrite(ledPin, HIGH);
14      Serial.print("Its Dark, Turn on the LED:");
15      Serial.println(ldrStatus);
16
17    }
18    else
19    {
20      digitalWrite(ledPin, LOW);
21      Serial.print("Its Bright, Turn off the LED:");
22      Serial.println(ldrStatus);
23    }
24  }
```

Given the following response function for the LDR, what is the light intensity that corresponds to the value of 900 in line 11?



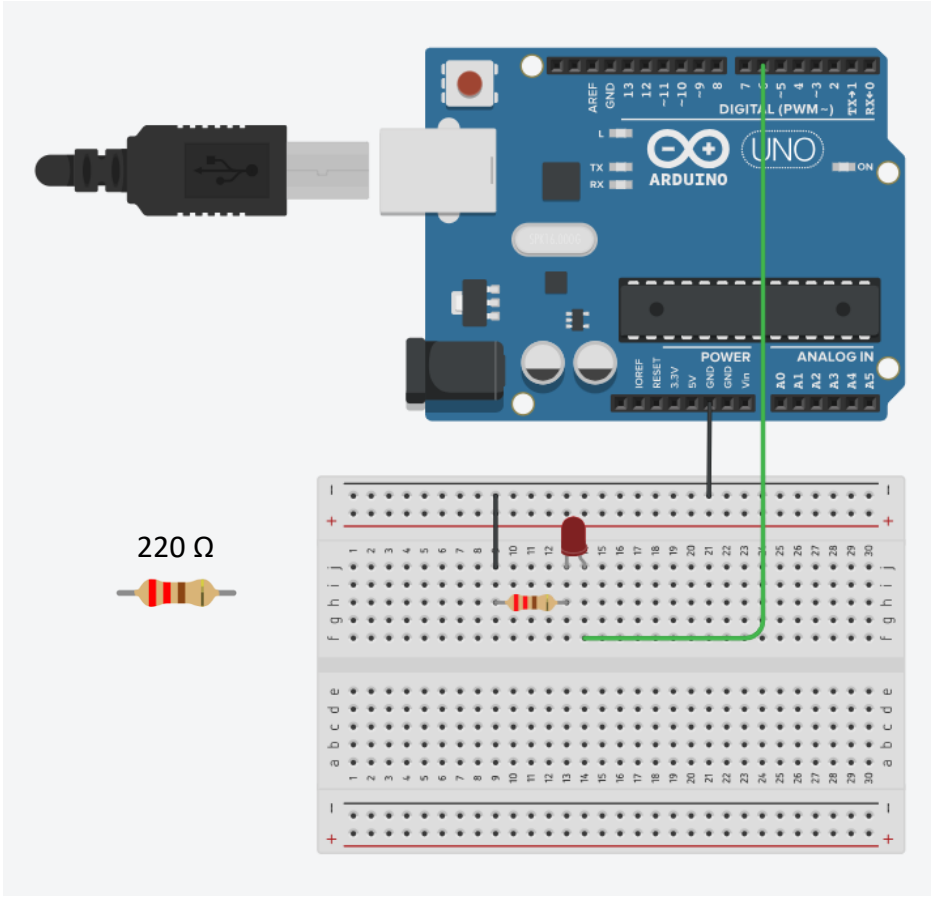


Microcontrollers

PWM – Pulse Width Modulation

Examples

```
1   //Initializing LED Pin
2   int led_pin = 6;
3
4   void setup() {
5     //Declaring LED pin as output
6     pinMode(led_pin, OUTPUT);
7   }
8
9   void loop() {
10    //Fading the LED
11    for(int i=0; i<255; i++){
12      analogWrite(led_pin, i);
13      delay(5);
14    }
15
16    for(int i=255; i>0; i--){
17      analogWrite(led_pin, i);
18      delay(5);
19    }
20
21  }
```

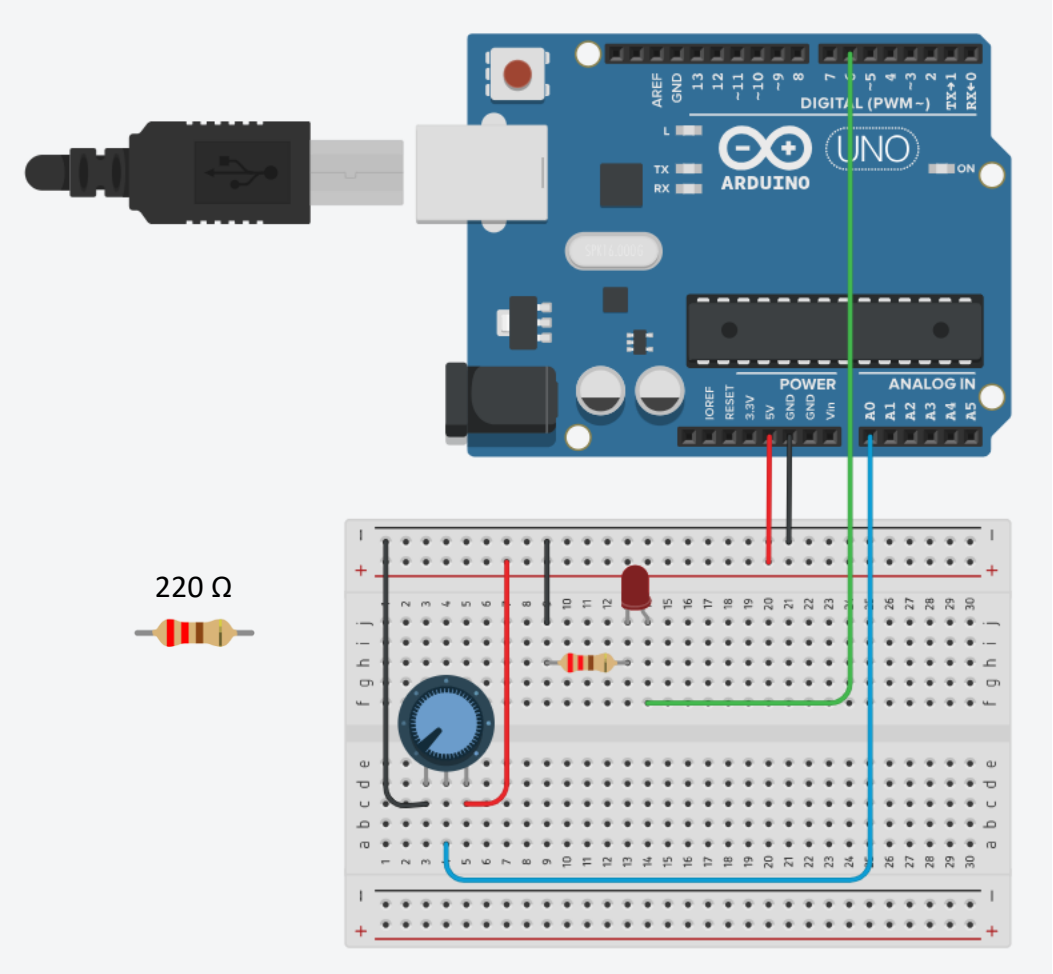


Microcontrollers

PWM – Pulse Width Modulation

Examples

```
1   int led_pin = 6;
2   int pot_pin = A0;
3   int output;
4   int led_value;
5   void setup() {
6     pinMode(led_pin, OUTPUT);
7   }
8   void loop() {
9     //Reading from potentiometer
10    output = analogRead(pot_pin);
11    //Mapping the Values between 0 to 255 because we can
12    //give output from 0 -255 using the analogwrite funtion
13    led_value = map(output, 0, 1023, 0, 255);
14    analogWrite(led_pin, led_value);
15    delay(5);
16  }
```



Microcontrollers

Exercise

A car's parking assistance system consists of a distance sensor mounted on the rear of its body and four LED indicator lights (green, yellow, orange, and red) mounted on the interior console. The indicator lights are activated according to the measured distance between the vehicle and a possible obstacle in accordance with the following cases:

- Green LED: distance greater than 60cm
- Yellow LED: distance between 60cm and 40cm
- Orange LED: distance between 40cm and 20cm
- Red LED: distance less than 20cm

The user can also dim the light intensity of the above LEDs through a potentiometer.

You need to:

- Implement the above circuit in a simulation package through the Arduino microcontroller.
- Compose a report including the developed circuit and the respective code.