Boosting programming skills with microcontrollers

Developed by Mirela TIBU

"Grigore Moisil" Theoretical Highschool of Informatics , Iasi, Romania

A Trainers Toolkit To Foster STEM Skills Using Microcontroller Applications

Co-funded by the Erasmus+ Programme of the European Union

Project No. 2019-1-RO01-KA202-063965

This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Contents

Computer Science

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Aim

Expressing a creative way of thinking, in structuring and solving problems

Forming habits to use specific algorithmic computer concepts and methods in approaching a variety of problems

Manifesting attitudes towards science and knowledge

Manifesting initiative and willingness to address various tasks

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Description

- ITC approach - Microprocessor VS Microcontroller
  - ✓ identifying areas where computers / embedded systems are used in daily life
  - ✓ describing hardware architecture for a computer and an embedded system
  - ✓ comparing microprocessors and microcontrollers features
- Programming approach - Understanding Programming Concepts by using Microcontrollers applications
  - ✓ structured programming statements – decisions, loops (IF, WHILE, FOR)
  - ✓ declaration & call of void and non-void functions
  - ✓ using arrays in applications
  - ✓ analyzing the functionality of Arduino devices to recognize the the execution steps of programming statements

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Learning Goals

- Identifying computer applications in social life - awareness of the impact of embedded systems in everyday life
  - Identifying the similarities and differences between a microprocessor and a microcontroller in the architecture of computing and embedded systems
  - Practicing the implementation of the elements of structured programming - decisions, loops, functions; representation and use of array data
  - Visualizing the effect of executing various program sequences through microcontroller-based devices

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Learning Methodologies

- Explanation
- Demonstration
- Conversation
- Algorithmization
- Implementations

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Target Group

Highschool students – 9-10th grades

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

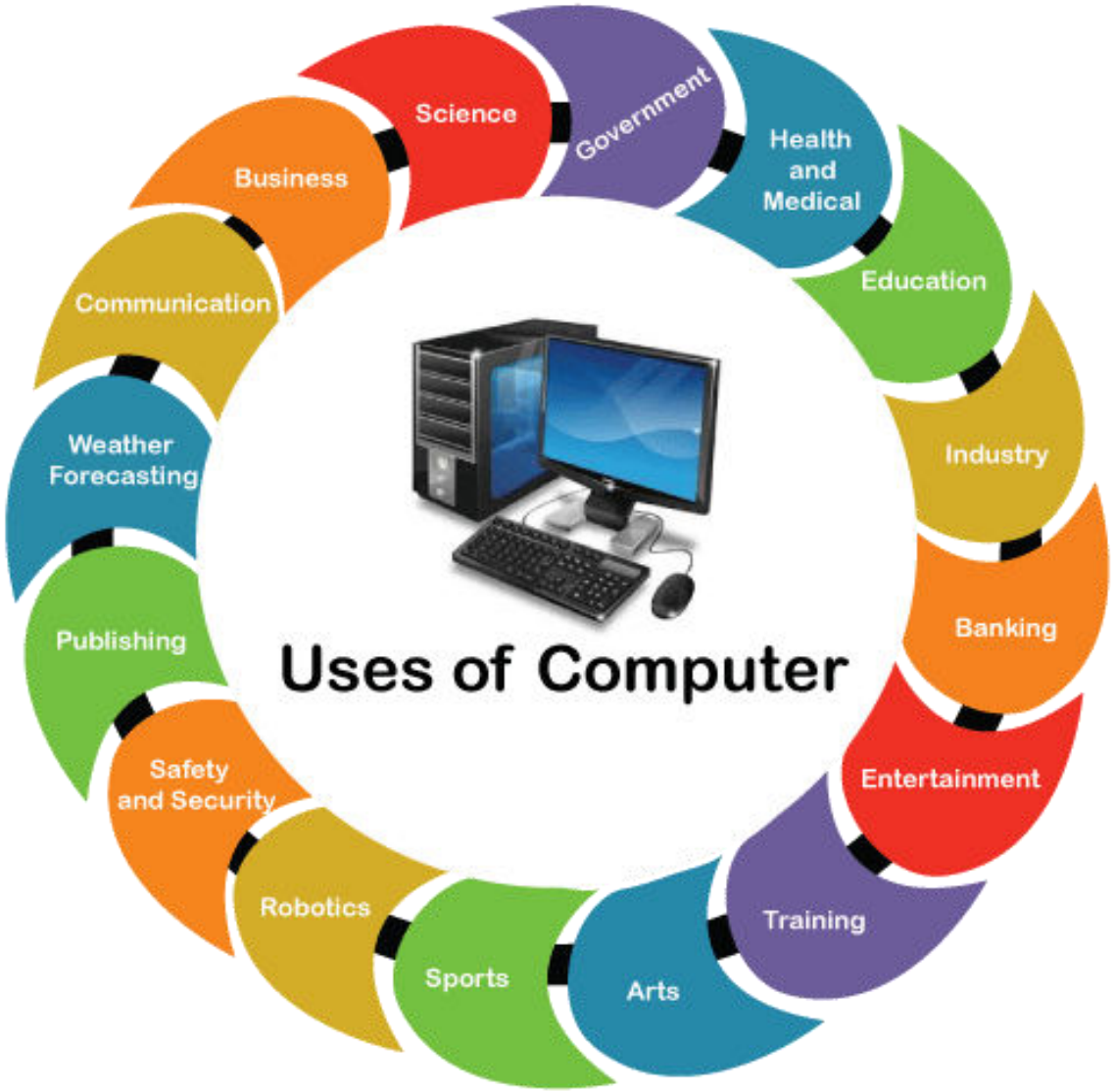
Co-funded by the Erasmus+ Programme of the European Union

Computers and Embedded Systems

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Computers and Daily Life



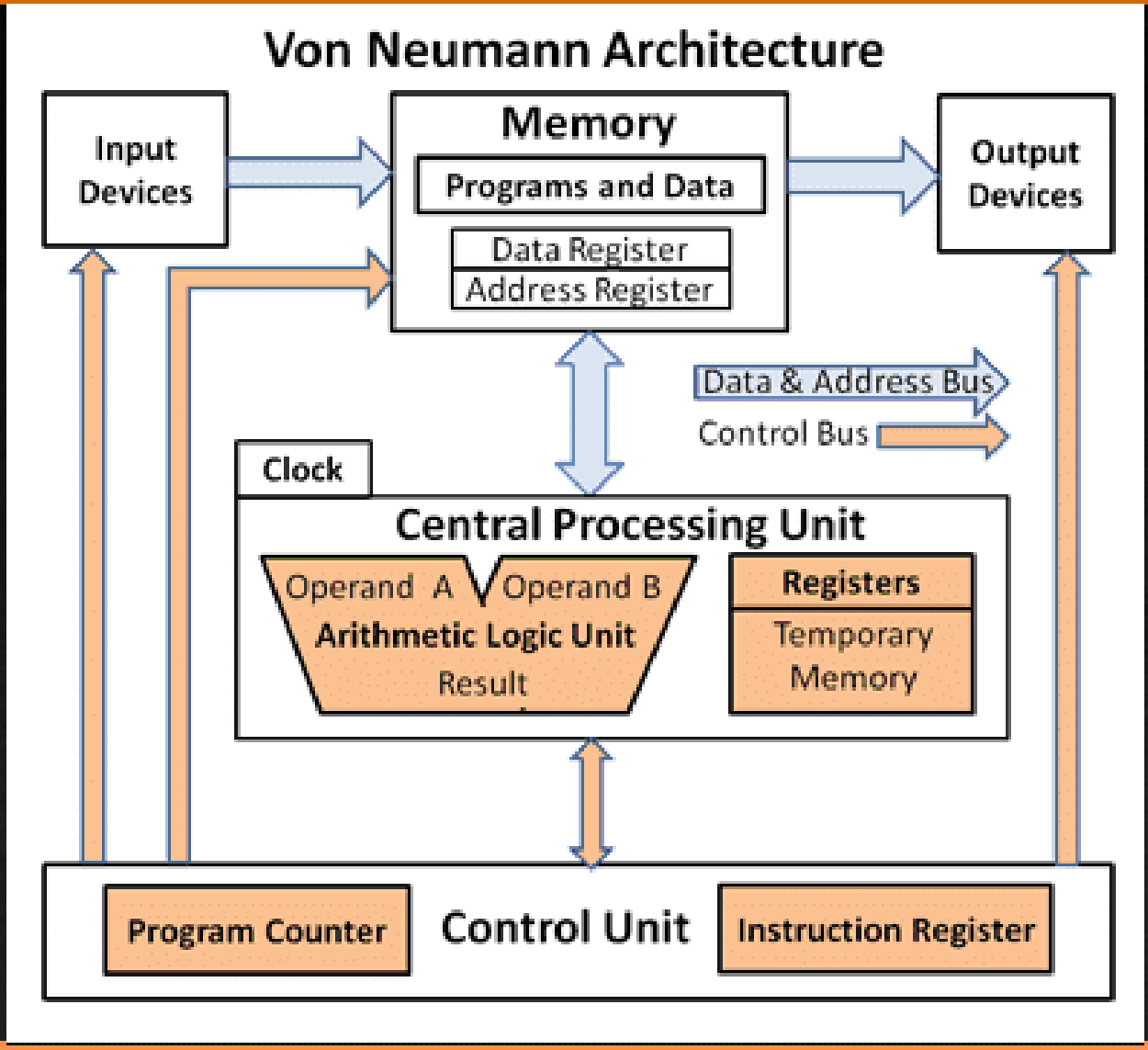
- Computers are part of our daily life.
- Computer = Hardware + Software
  - ✓ Hardware - physical components
  - ✓ Software - programs, procedures, and routines that tells a computer what to do
- WHERE do we use computers?

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Computer Hardware Architecture

- ✓ CPU = Microprocessor - “the brain” of our computer – makes all the arithmetical and logical operations (ALU) and control all system activities
- ✓ Memory Unit – store data and programs
  - RAM – Random Access Memory
  - ROM – Read Only Memory
- ✓ Input/Output devices



Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

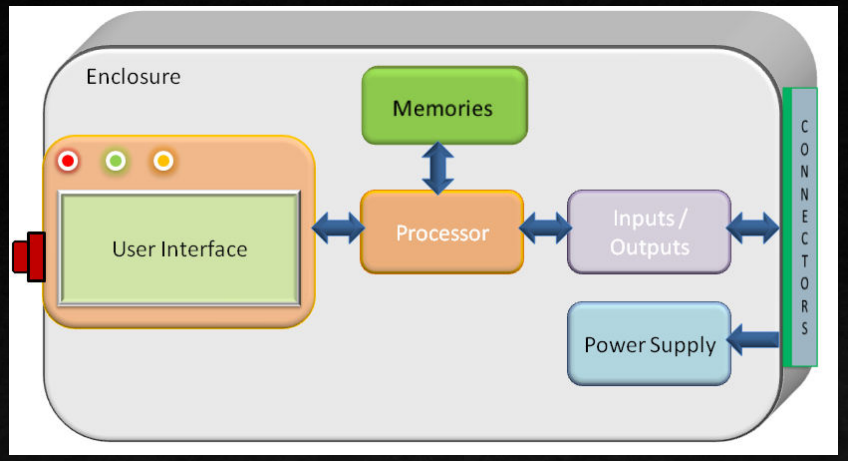
Embedded Systems in Daily Life

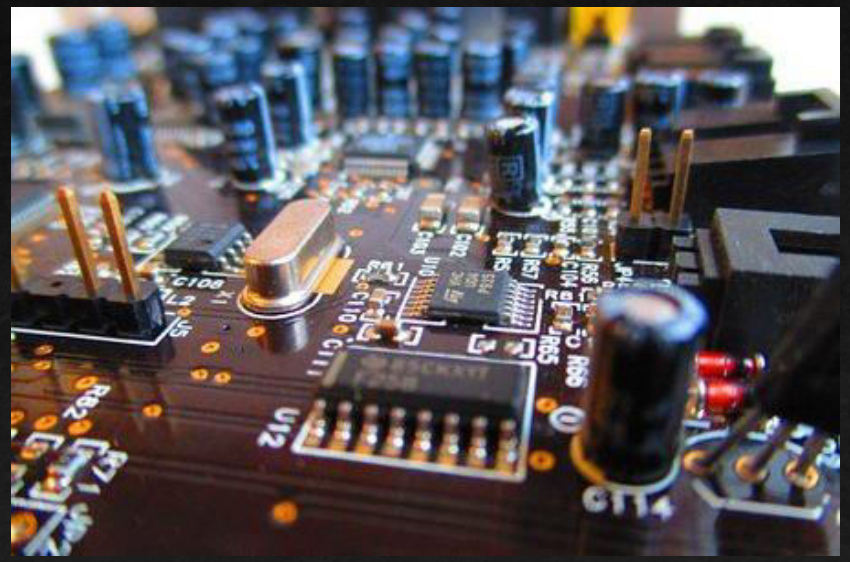
- An **embedded system** is a special purpose computer that is used inside of a device
- is based on microcontroller which is a chip optimized to control electronic devices; it is stored in a single integrated circuit, dedicated to perform a particular task and execute one specific application
- WHERE do we use embedded systems?

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

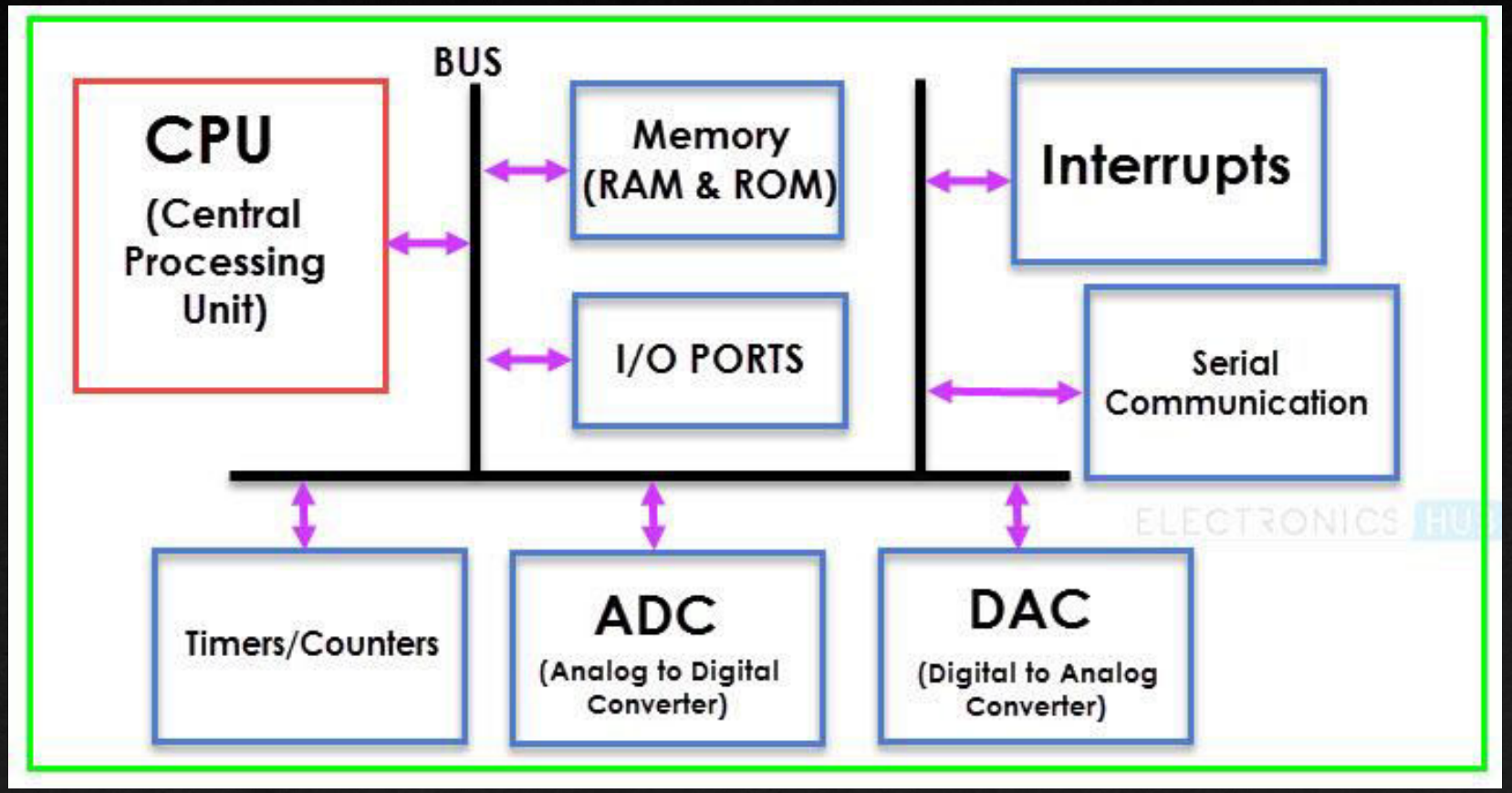
Co-funded by the Erasmus+ Programme of the European Union

Embedded Systems Architecture





- CPU – is a microcontroller or a microprocessor



Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Which one is better?

Microprocessor	Microcontroller
is the heart of Computer system.	the heart of an embedded system
it is only a processor, so memory and I/O components need to be connected externally	has a processor along with internal memory and I/O components
memory and I/O has to be connected externally, so the circuit becomes large	Memory and I/O are already present, and the internal circuit is small
it can be used in compact systems	it is used in compact systems
cost of the entire system is high	cost of the entire system is low
due to external components, the total power consumption is high. it is not ideal for the devices running on stored power like batteries.	as external components are low, total power consumption is less. So it can be used with devices running on stored power like batteries
most of them do not have power saving features	most of the them offer power-saving mode
mainly used in personal computers	mainly used in embedded systems.
are based on Von Neumann model	are based on Harvard architecture

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Which one is better?

have a smaller number of registers, so more operations are memory-based	have more register, so the programs are easier to write
is a central processing unit on a single silicon-based integrated chip	is a byproduct of the development of microprocessors with a CPU along with other peripherals
has no RAM, ROM, Input-Output units, timers, and other peripherals on the chip	has a CPU along with RAM, ROM, and other peripherals embedded on a single chip
uses an external bus to interface to RAM, ROM, and other peripherals	uses an internal controlling bus
Microprocessor-based systems can run at a very high speed because of the technology involved	Microcontroller based systems run up to 200MHz or more depending on the architecture
is used for general purpose applications that allow you to handle loads of data	is used for application-specific systems
It's complex and expensive, with a large number of instructions to process	It's simple and inexpensive with less number of instructions to process

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

IF statement, declaration & call of void functions and const definition

Task: Program a device capable to read the state of a potentiometer (an analog input) and turns on an LED only if the potentiometer.
It must print the analog value regardless of the level.

```
const int analogPin = A0;
   // pin that the sensor is attached to
const int ledPin = 13;
   // pin that the LED is attached to
const int valmin = 400;
   // an arbitrary valmin level

void setup() {
   // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
   // initialize serial communications:
  Serial.begin(9600); }
```

```
void loop() {
  // read the value of the potentiometer:
  int analogValue = analogRead(analogPin);

  // if the analog value is high enough,
  // turn on the LED:
  if (analogValue > valmin) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
  // print the analog value:
  Serial.println(analogValue);
  delay(1);    // delay in between reads
}
```


Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Understanding Programming Concepts by using Microcontroller applications

Variable definition

```
<typeVar> nameVar [= value];
```

declares a variable of a specific type, witch will determin the size of the values, the length and type of memory representation

Usual types of variables used in Arduino apps

int = numeric type for variables/constants;
it is represented on 4 bytes and can store values between aprox. $-2*10^9 \cdots 2*10^9$

bool = boolean type for variables/constants;
it is represented in 1 byte and can store values false (0) and true (1)

Constant definition

```
const <typeConst> nameConst = value;
```

sets a constant of typeConst with specific value

Decision Structure

```
if (Condition) {
  instructions_A
   // do instructions_A if the
   // Condition is true
} else {
 instructions_B
  // do instructions_A if the
  // Condition is false
}
```

void Functions – declaration

```
void nameFunction(list of formal parameters)
{ declaration of local variables
  instructions
}
```


Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Understanding Programming Concepts by using Microcontroller applications

void Functions – declaration

```
void nameFunction(formal parameters)
{ declaration of local variables
  instructions
}
```

non-void Functions - declaration

```
resultType nameFunction(formal parameters)
{ declaration of local variables
  instructions
  return expression;  //
}
```

where formal parameters is a list of types and names of parameters used in function instructions

Call of a function

```
nameFunction(list of actual parameters)
```

- void functions - The call is an instruction
- non-void function – The call is an oprerand in expression with the same type as the resultType

! formal and actual parameters must have same type, number and must be in the same order

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Arduino specific functions

setup()

`void function` - called when a sketch starts, and will only run once, after each powerup or reset of the Arduino board similar with main(). Use it to initialize variables, pin modes, start using libraries, etc.

loop()

`void function` - loops consecutively, allowing Arduino program to change and respond
It is called after `setup()` function, which initializes and sets the initial values.

pinMode(pin, mode)

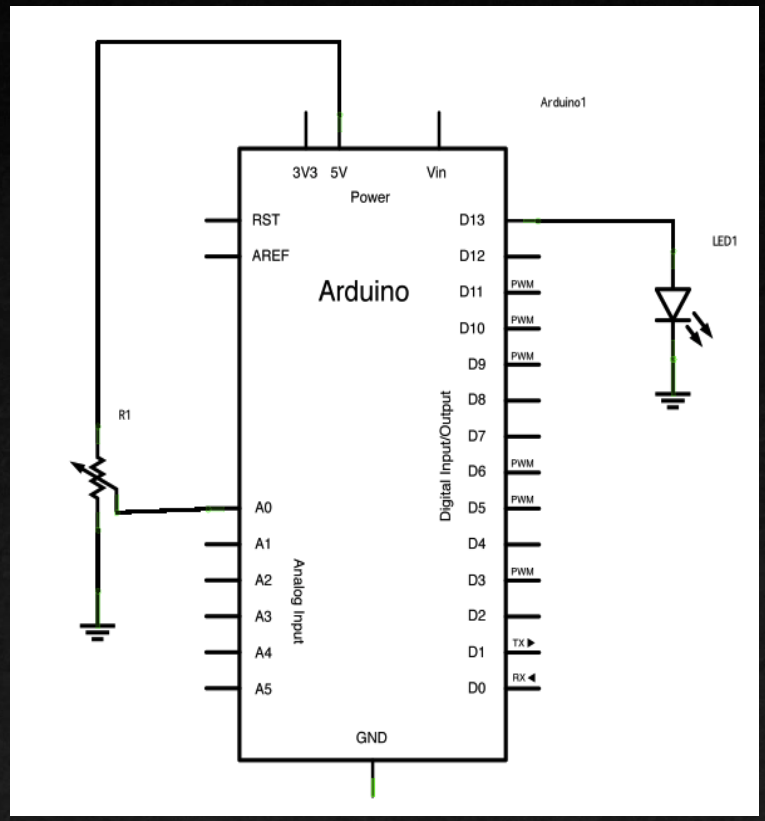
`void function with parametrers`

- `pin`: the Arduino pin number to set the mode off
- `mode`: INPUT, OUTPUT, or INPUT_PULLUP

delay(milisec)

`void function with parametrers`

- `milisec`: number of miliseconds to pause the program



electrical schema of the Arduino device

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Arduino specific functions

digitalWrite(pin, value)

void function with parametrers

- pin: the Arduino pin number.
- value: HIGH or LOW

digitalRead(pin)

function with parametrers

- pin: the Arduino pin num
- return value: HIGH or LOW

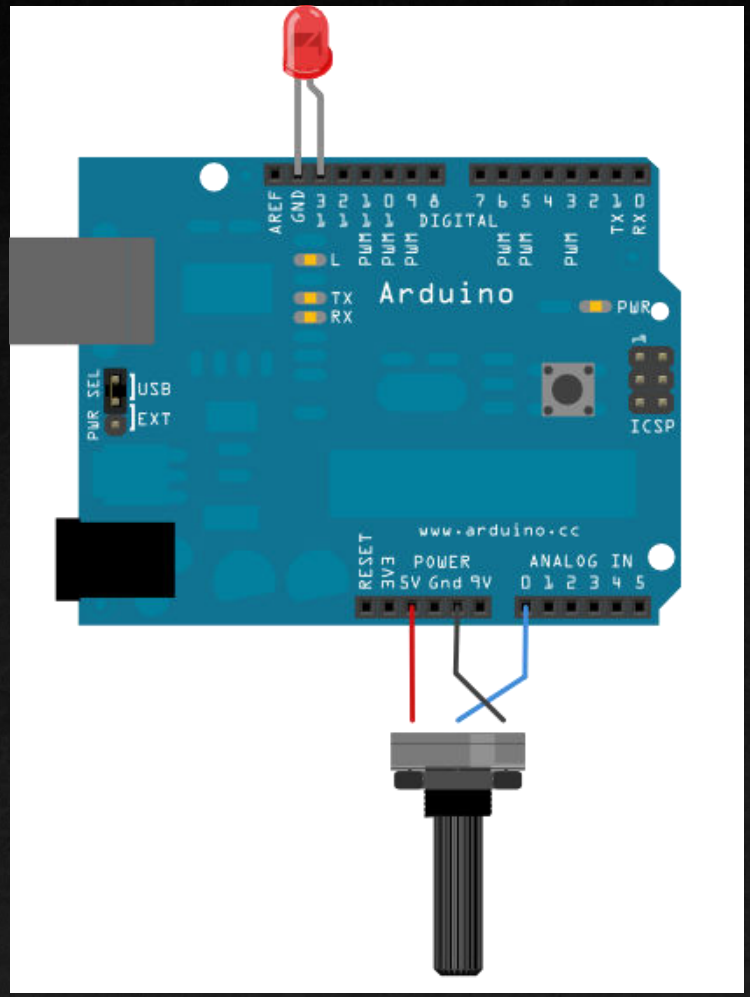
analogRead(pin)

int function with parametrer

- pin: the name of the analog input pin to read from (A0 to A5 on most boards)
- Returns the analog reading on the pin.

Arduino Pin Levels Constants HIGH and LOW

pin	INPUT	OUTPUT
HIGH	voltage > 3.0V	5V
LOW	voltage > 3.0V	0V



Arduino device

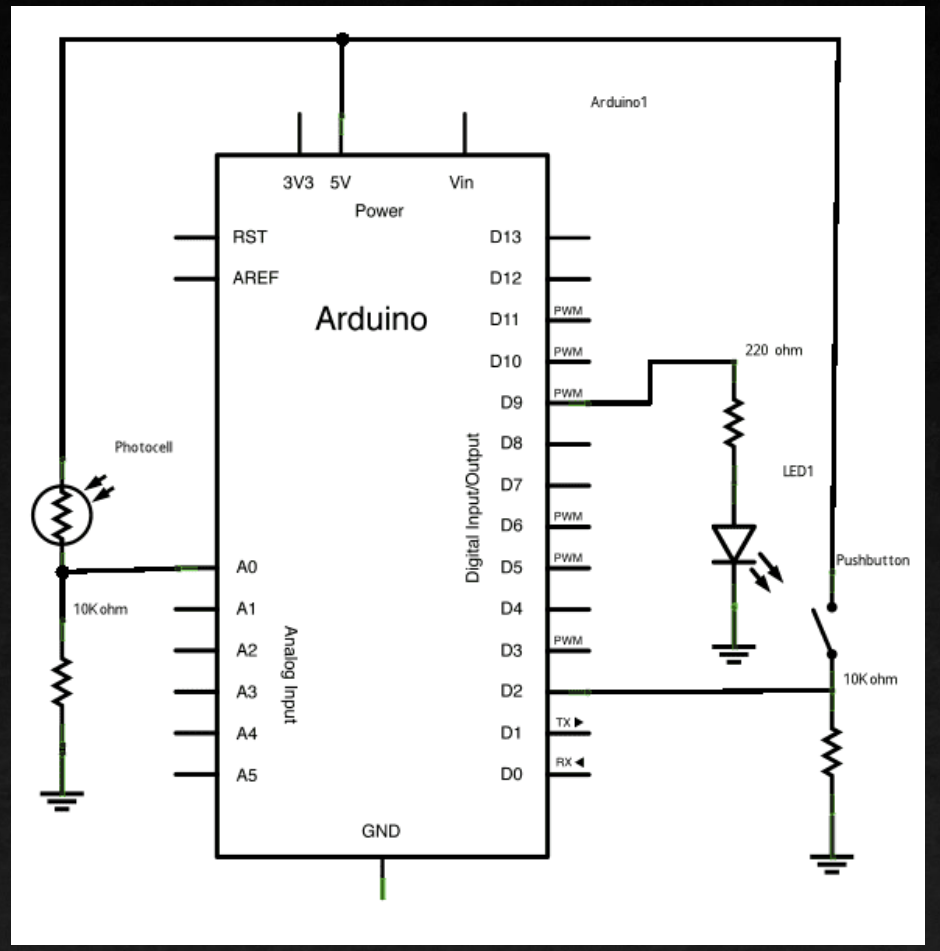
Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

WHILE statement

Task: Program a device capable to read for five seconds the sensor input and calibrate by defining the minimum and maximum of expected values for the readings taken during the loop.

```
const int sensorPin = A0;
     // pin that the sensor is attached to
const int ledPin = 9;
     // pin that the LED is attached to
     // variables:
int sensorValue = 0;
    // the sensor value
int sensorMin = 1023;
    // minimum sensor value
int sensorMax = 0;
    // maximum sensor value
```



Schema for Arduino device

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

WHILE statement

```
void setup() {
  // turn on LED to signal the start
  // of the calibration period:
     pinMode(13, OUTPUT);
     digitalWrite(13, HIGH);
  // calibrate during the first five seconds
  while (millis() < 5000) {
       sensorValue = analogRead(sensorPin);
       if (sensorValue > sensorMax){
           sensorMax = sensorValue;
       }
       if (sensorValue < sensorMin){
          sensorMin = sensorValue;
       }
   }
  digitalWrite(13, LOW);    // signal the end of the calibration period
}
```


Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Determin the Minimum and Maximum value from a set of values

Algorithm

Step 1. Set the variables for sensorMin with the maximum value possible and sensor Max with the minimum value posibile

Step 2. Compare current value with sensorMinand, if it is smaller, update sensorMin

Step 3. Compare current value with sensorMax and, if it is greater, update sensorMax

WHILE statement

```
while (Condition) {
  instructions_A
}
```

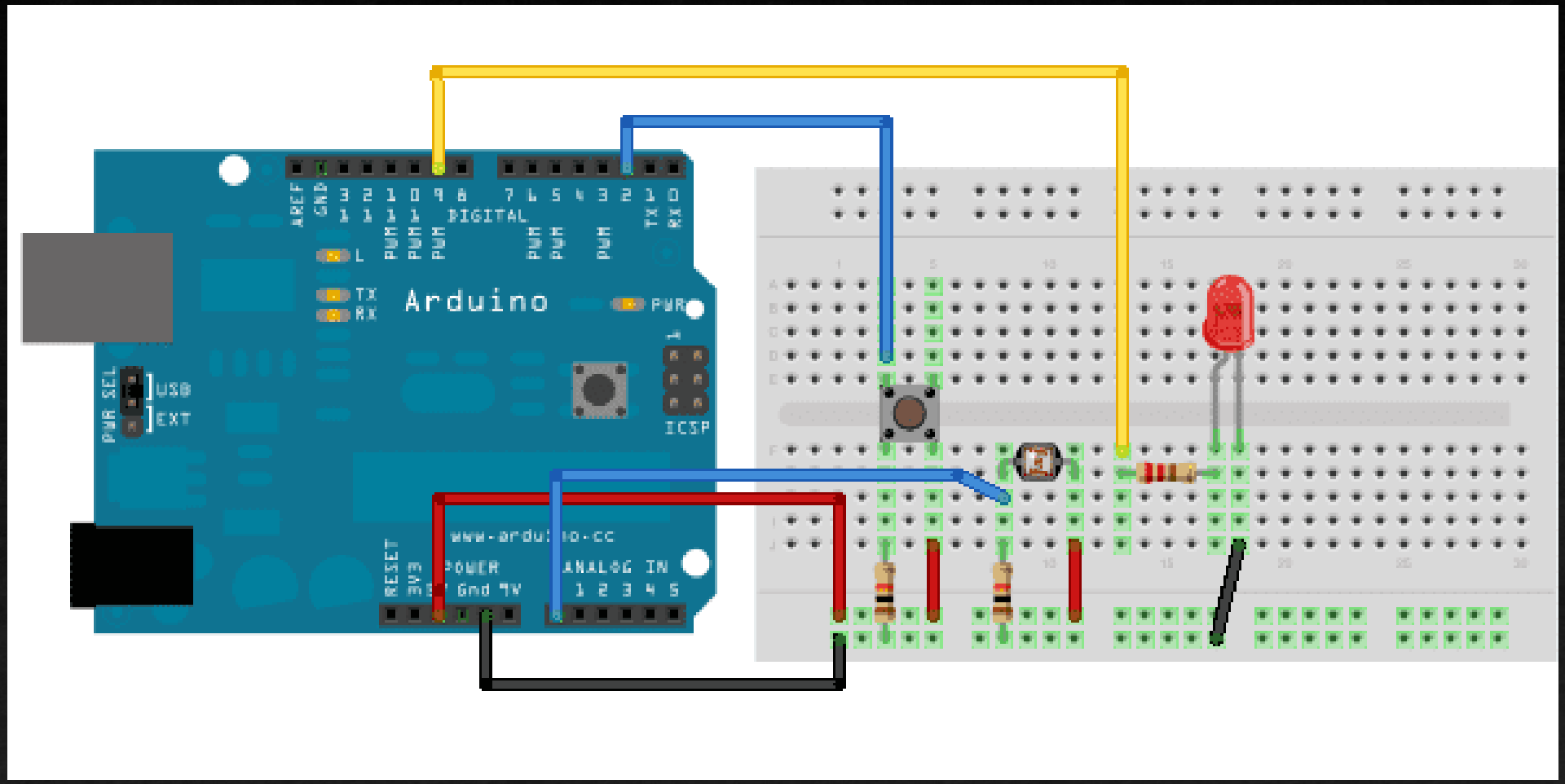
Execution

Step 1. The Condition is evaluated
Step 2. If the Condition is True
2.1. Instructions A will be executed
2.2. Go to Step 1.
If Condition is False, the program execution leave the loop

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Understanding Programming Concepts by using Microcontroller applications



Arduino device explaining WHILE

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

FOR statement and ARRAYs manipulation

```
int timer = 100;
   // The higher the number,
   // the slower the timing.
int ledPins[] = { 2, 7, 4, 6, 5, 3};
   // an array of pin numbers to which LEDs are attached
int pinCount = 6;
  // the number of pins (the length of the array)

void setup() {
  // the array elements are numbered from 0 to (pinCount-1)
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 0; thisPin < pinCount; thisPin++) {
    pinMode(ledPins[thisPin], OUTPUT);
  }
}
```

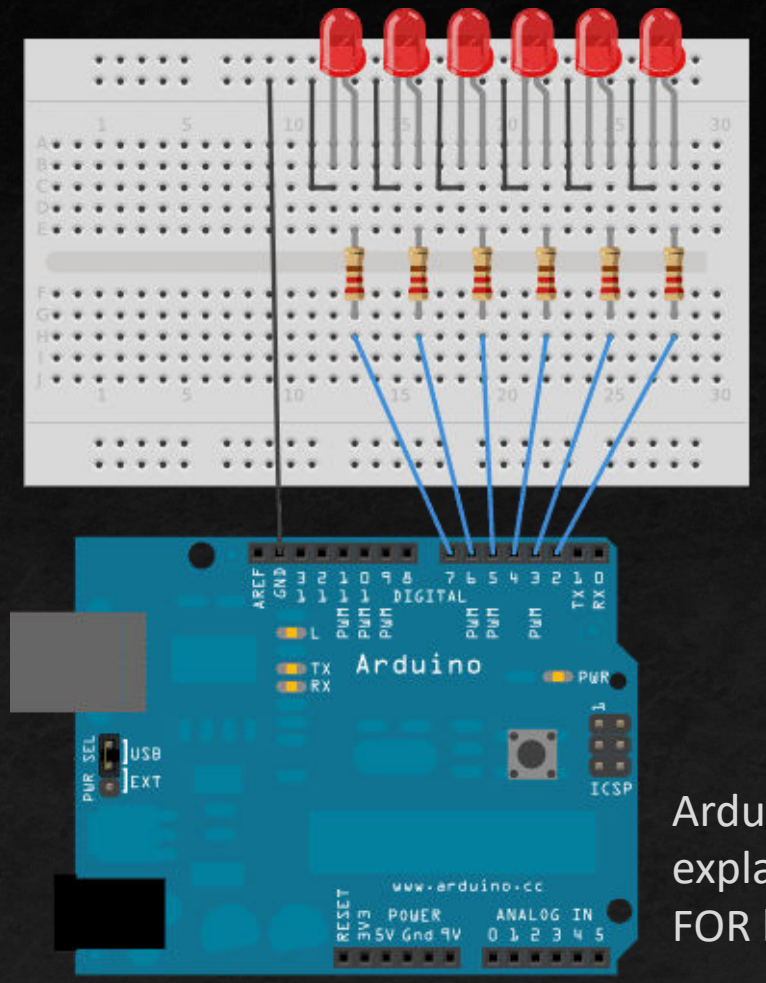
Task: Program a device capable to light up a series of LEDs attached to pins whose numbers are neither contiguous nor necessarily sequential. To do this is, pin numbers will be store in an ARRAY and then use FOR loops to iterate over the array.

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

FOR statement and ARRAYs manipulation

```
void loop() {
// loop from the lowest pin to the highest:
 for (int thisPin = 0; thisPin < pinCount; thisPin++) {
   digitalWrite(ledPins[thisPin], HIGH);
      // turn the pin on:
   delay(timer);
   digitalWrite(ledPins[thisPin], LOW);
      // turn the pin off:
 }
      // loop from the highest pin to the lowest:
 for (int thisPin=pinCount-1; thisPin >= 0; thisPin--){
      // turn the pin on:
    digitalWrite(ledPins[thisPin], HIGH);
    delay(timer);
      // turn the pin off:
    digitalWrite(ledPins[thisPin], LOW);
 }
}
```

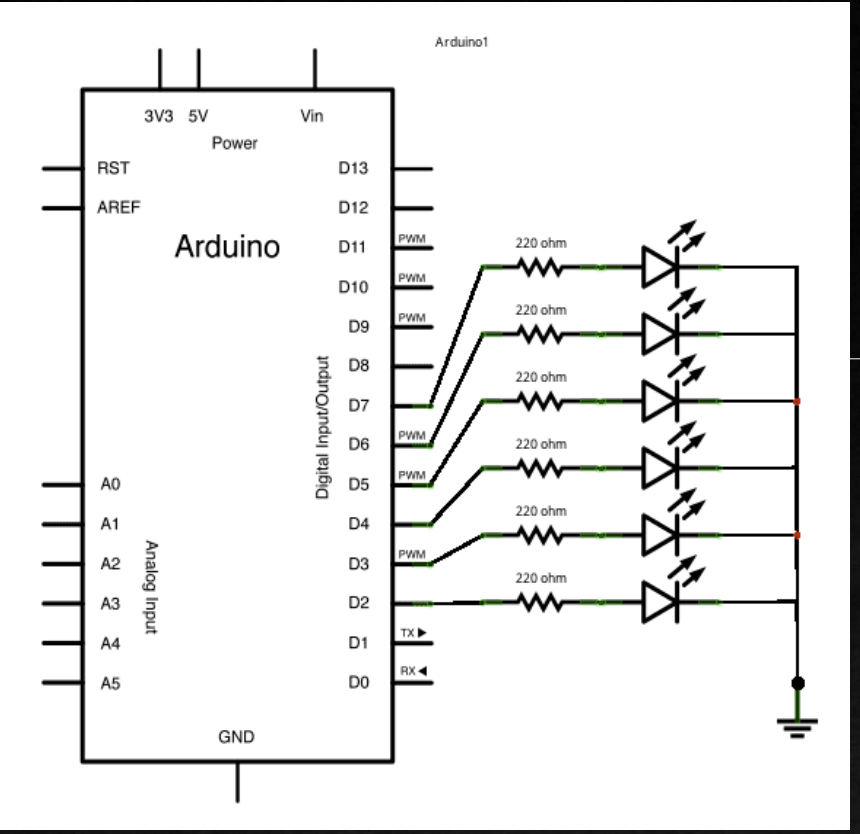


Arduino device explaining FOR loop

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Understanding Programming Concepts by using Microcontroller applications



Schema for Arduino device

FOR statement

```
for(int counter = initialVal; counter <= finalVal; counter++) {
     instructions_A
}
```

Execution

Step 1. The counter is set with initialValue

Step 2. If counter <= finaValue is True

- 2.1. Instructions A will be executed
- 2.2. counter is increased with 1
- 2.3. Go to Step 2

Step 3. If counter <= finaValue is False, the program execution leave the loop

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Organizing data in ARRAYs

Array = a collection of data with the same type, organized in a contiguos memory zone and refered with a single name, witch is a pointer (memory adress) of the first element in the array.

Declaration:

```
valuesType arrayName[numberOfElements];
```

Initialization:

- along with the declaration

```
int ledPins[] = { 2, 7, 4, 6, 5, 3};
```

- by assignation

```
digitalWrite(ledPins[thisPin], HIGH);
```

- by reading values from input

Refering to a specific value from the array

A[expressionIndex]

expressionIndex is an integer from [0,count-1], indicating the position of the element in array

Parsing the ARRAY in order to analyze and process its elements:

left-rigt

```
for (int it = 0; it < count; it++)
    process(A[it]);
```

rigt-left

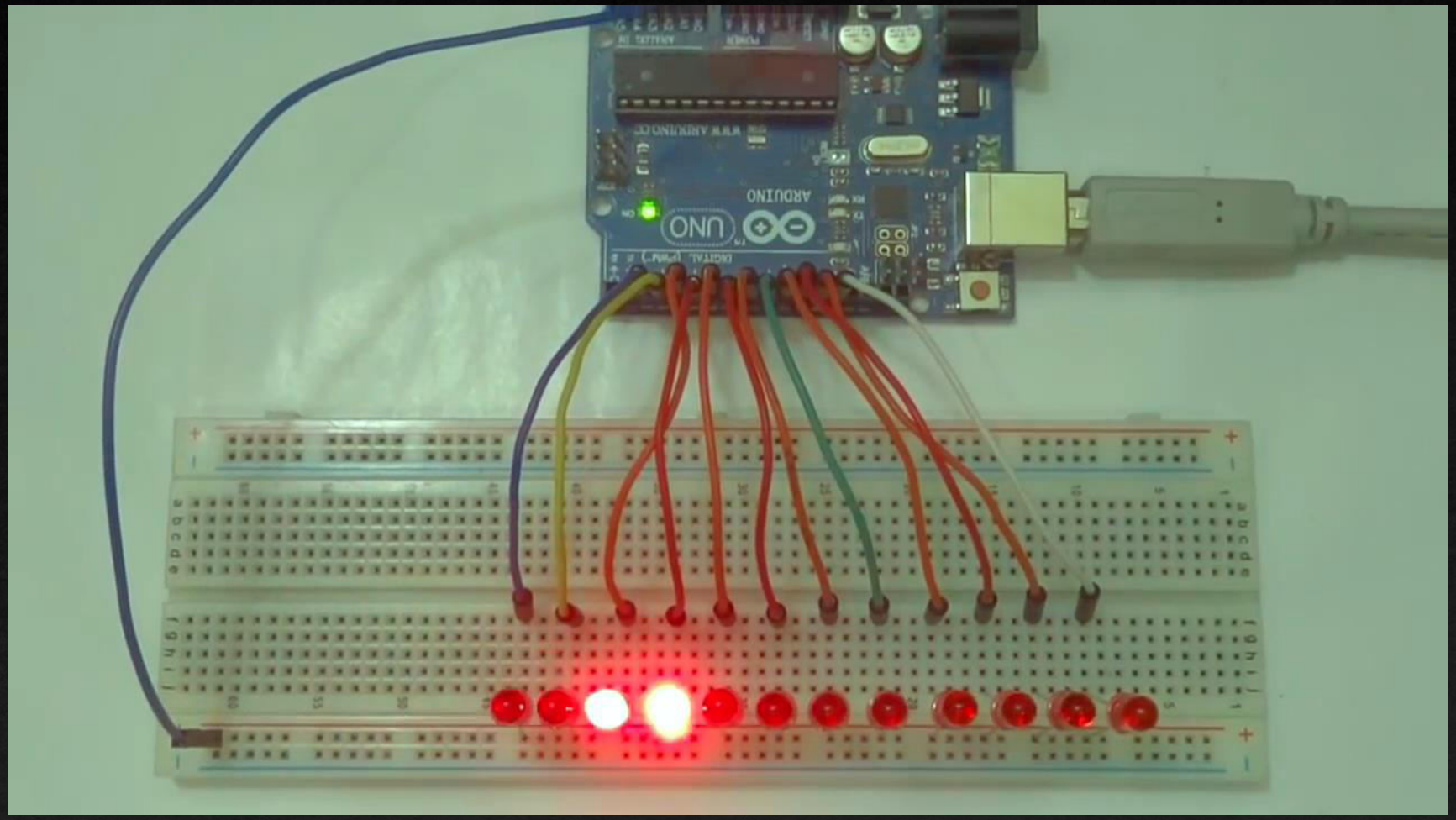
```
for (int it = count - 1; it >= 0; it--)
    process(A[it]);
```

where count is the number of array elements and it is an index used for parsing the array

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Understanding Programming Concepts by using Microcontroller applications



Learning FOR loops and ARRAYs with Arduino device

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Scientific Areas Covered

Hardware architecture for Computers and Embedded Systems

Structured Programming – data types, statements (IF, WHILE, FOR), user-defined functions

Programming Microcontroller Based devices (ex. Arduino)

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Assessment

- multiple choice test
- mini-project in a team of 2-3 students - programming Arduino devices that:

✓ describe the operation of other specific instructions in structured programming

✓ to apply in real situations - for example, the operation of an RGV traffic light

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union

Bibliography

Webography

- https://creativecommons.org/2008/10/22/wired-on-arduino-and-open-source-computing/
- https://www.arduino.cc/en/Tutorial/BuiltInExamples/WhileStatementConditional
- https://www.arduino.cc/en/Tutorial/BuiltInExamples/ForLoopIteration
- https://commons.wikimedia.org/wiki/Category:Fritzing
- https://www.arduino.cc/en/Tutorial/BuiltInExamples/ForLoopIteration
- https://commons.wikimedia.org/wiki/Category:Arduino_projects

Project No. 2019-1-RO01-KA202-063965
This project has been funded with support from the European Commission. The content reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Co-funded by the Erasmus+ Programme of the European Union