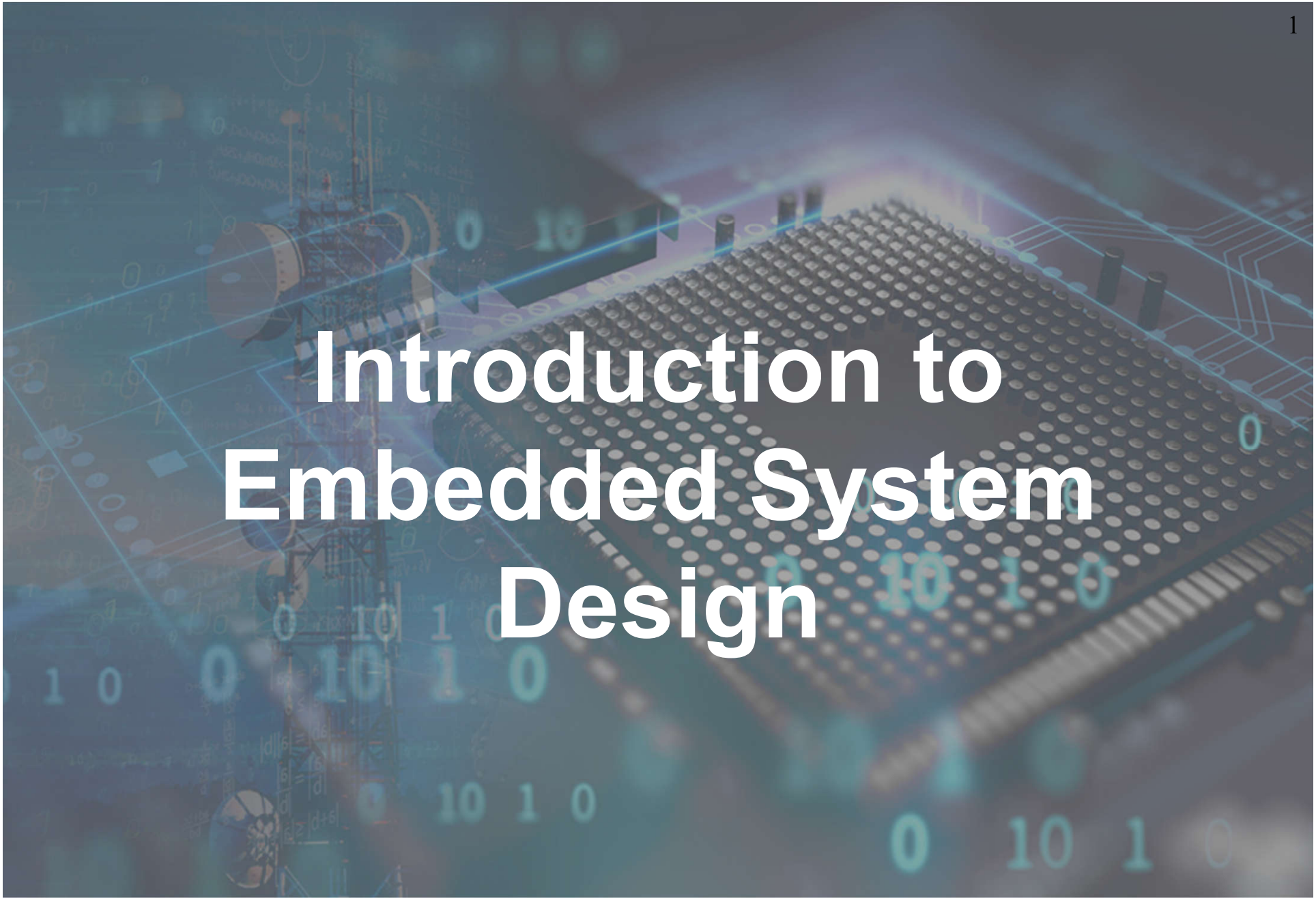


Introduction to Embedded System Design

Introduction to Embedded System

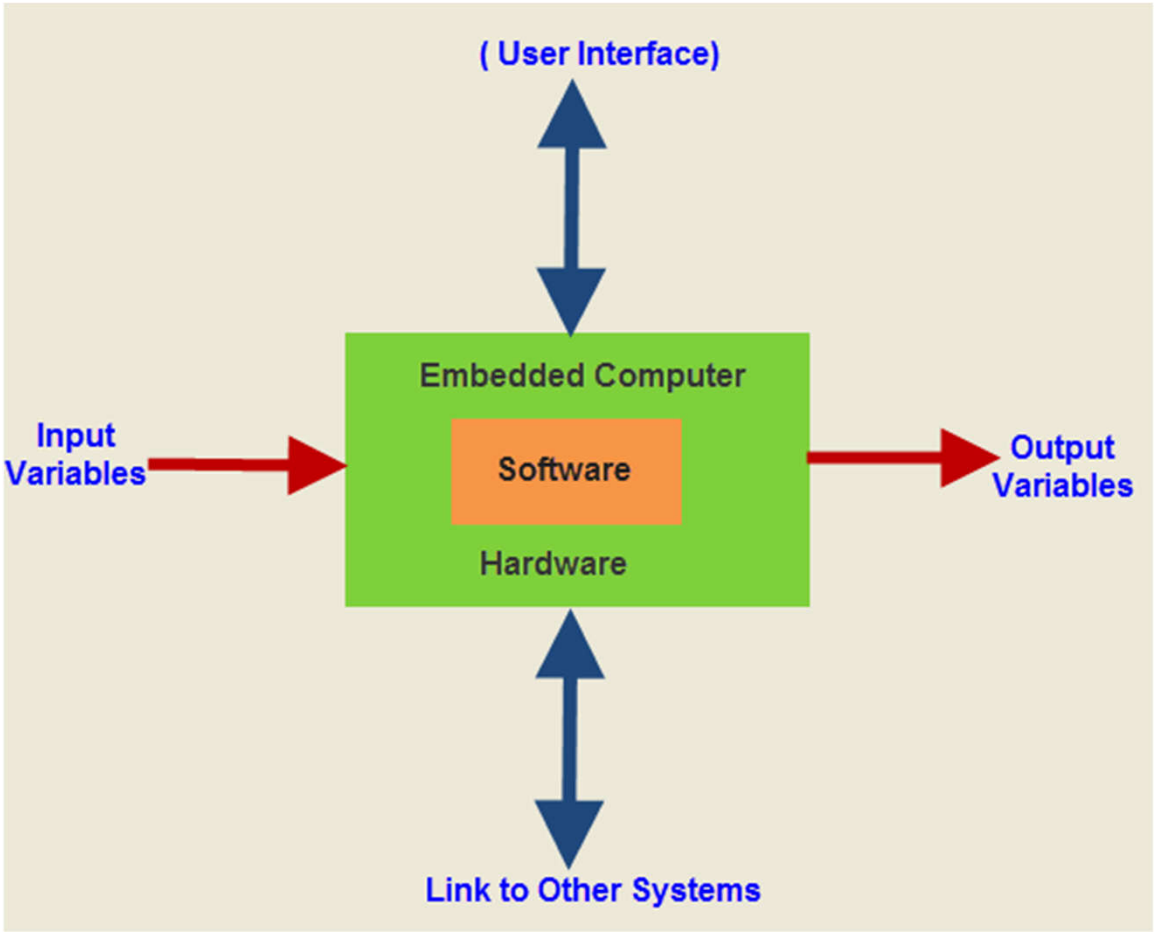
Contents

Objectives

- In this tutorial you will learn:
  - To understand basic concepts of Embedded Systems
  - To appreciate why it is important to learn Embedded Systems
  - To become familiar with different applications of Embedded Systems.
  - To become familiar with the advantages and disadvantages of Embedded Systems.
  - To become aware of the characteristics of an Embedded Systems.
  - To understand the types of Embedded Systems.
  - To become aware of main elements of Embedded Systems.
  - To appreciate the history of Embedded Systems.

What is Embedded Systems?

- Combination of hardware, software or firmware designed for a specific task



What is Embedded Systems?

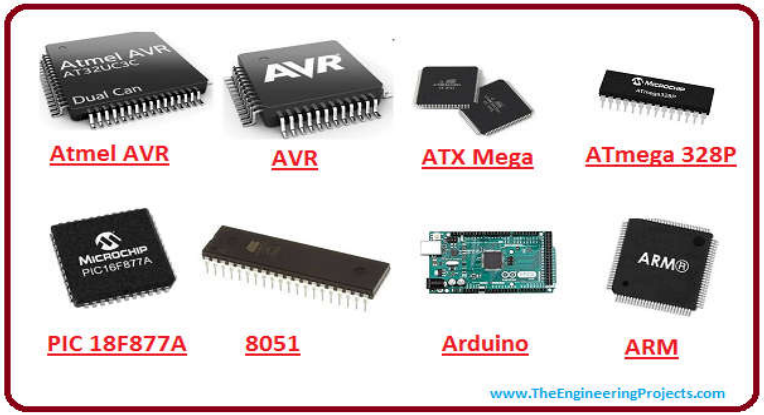
- Designed with finite functions within a larger system.

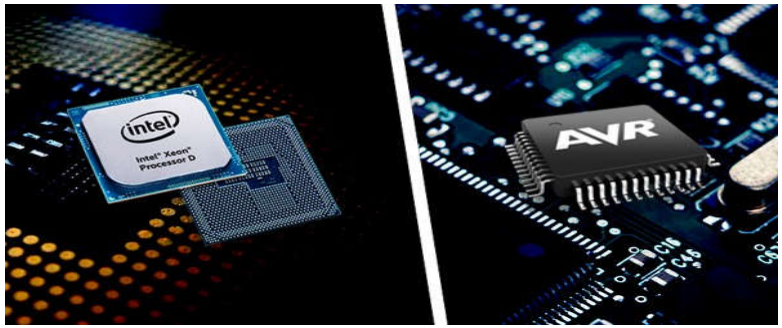
What is Embedded Systems?

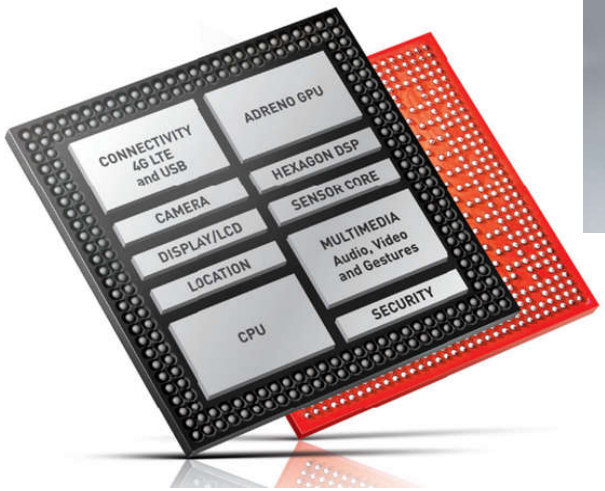
- Sometimes called "hidden computers".

What is Embedded Systems?

- Embedded systems can be programmable or fixed functionality.







What is Embedded Systems?

- It can be from no user interface to complex graphical user interfaces.

AMBIENT HUMIDITY / TEMPERATURE WIRELESS SENSOR

Why it is Important?

- Design engineers can optimize bill of materials of a product
- Reduce the size and cost of a product
- Increase the reliability and performance of the product
- Makes devices smarter and better connectivity
- Products are future proof and can be upgraded

Why it is Important?

- Design engineers can optimize bill of materials of a product
  - Microprocessor vs Microcontroller
  - FPGA vs Microcontroller
  - FPGA vs ASIC
  - Microcontroller vs SOC

Why it is Important?

- Reduce the size and cost of a product
  - Glue Logic vs Microcontroller/FPGA
  - Analog Circuit vs Microcontroller/PSOC
  - Relay Logic vs PLC

Why it is Important?

- Increase the reliability and performance of the product
  - Analog vs Digital

Why it is Important?

- Makes devices smarter and better connectivity
  - UART (RS232), I2C, RS485, WiFi, TCP/IP (LAN/WAN), Bluetooth
  - Internet of Things
  - Industry 4.0
  - Smart Things/Smart Sensors

Why it is Important?

- Products are future proof and can be upgraded
  - Embedded Systems are Programmable
  - Long Life Longevity
  - Microcontroller nowadays has big flash ROM
  - FPGAs has high logic gates count

Characteristics of an Embedded System

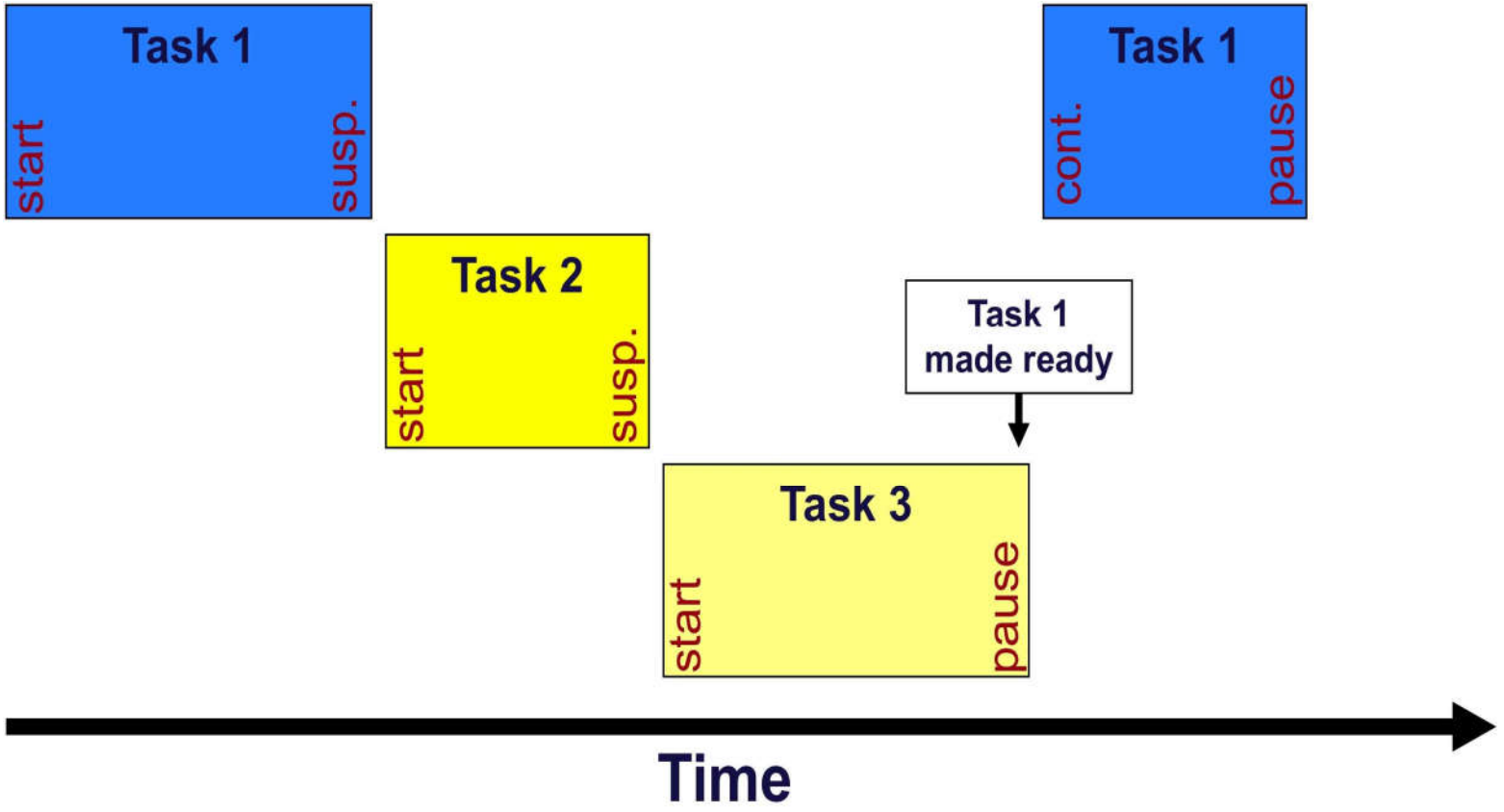
8 Characteristics of an Embedded Systems

1. Task Specific
2. Time Specific
3. Minimal User Interface
4. High Efficiency
5. High Reliability
6. High Stable
7. Low Cost
8. Requires Less Power

Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

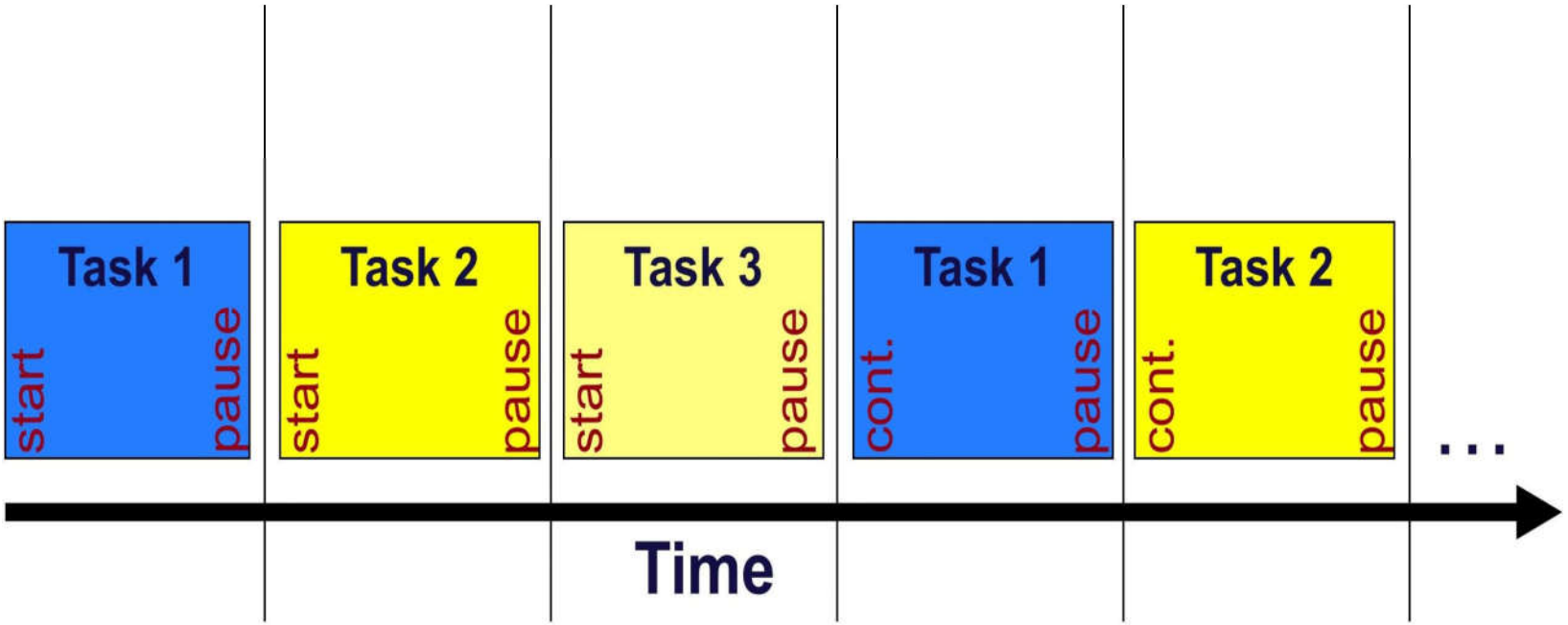
1. Task Specific



Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

2. Time Specific



Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

3. Minimal User Interface

Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

4. High Efficiency
   - Low Power Consumption

Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

5. High Reliability
   - Fault Tolerant

Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

6. High Stable

   - ADC, Timers, Baud rate, Clock, it should be precise

Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

7. Low Cost

- Price to Ratio performance

Characteristics of an Embedded System

8 Characteristics of an Embedded Systems

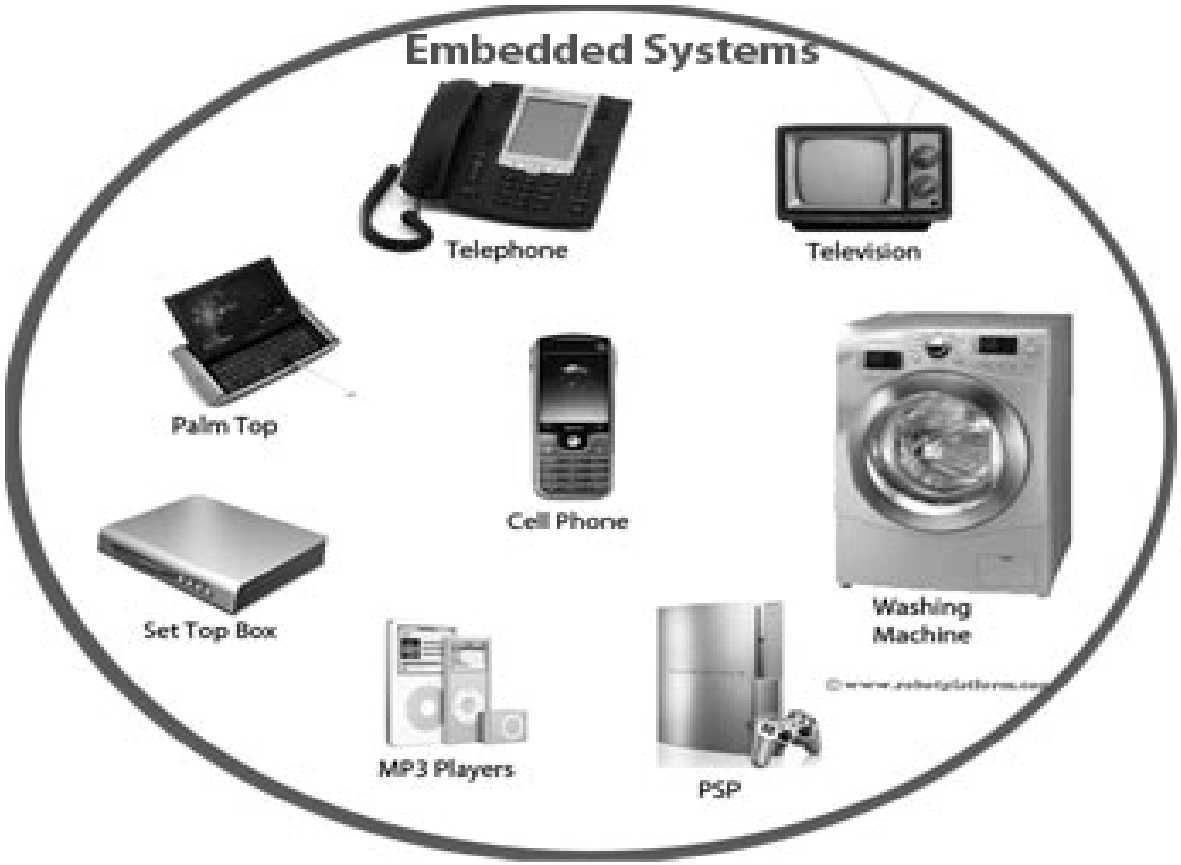
8. Requires Less Power
   - Battery powered/Long life

Applications of Embedded System

- Consumer Electronics
- Robotics
- Automotive Industries
- Industrial Machines
- Test Equipment
- Medical Equipment
- Military Applications
- Aerospace
- Communication Systems
- Navigation Systems

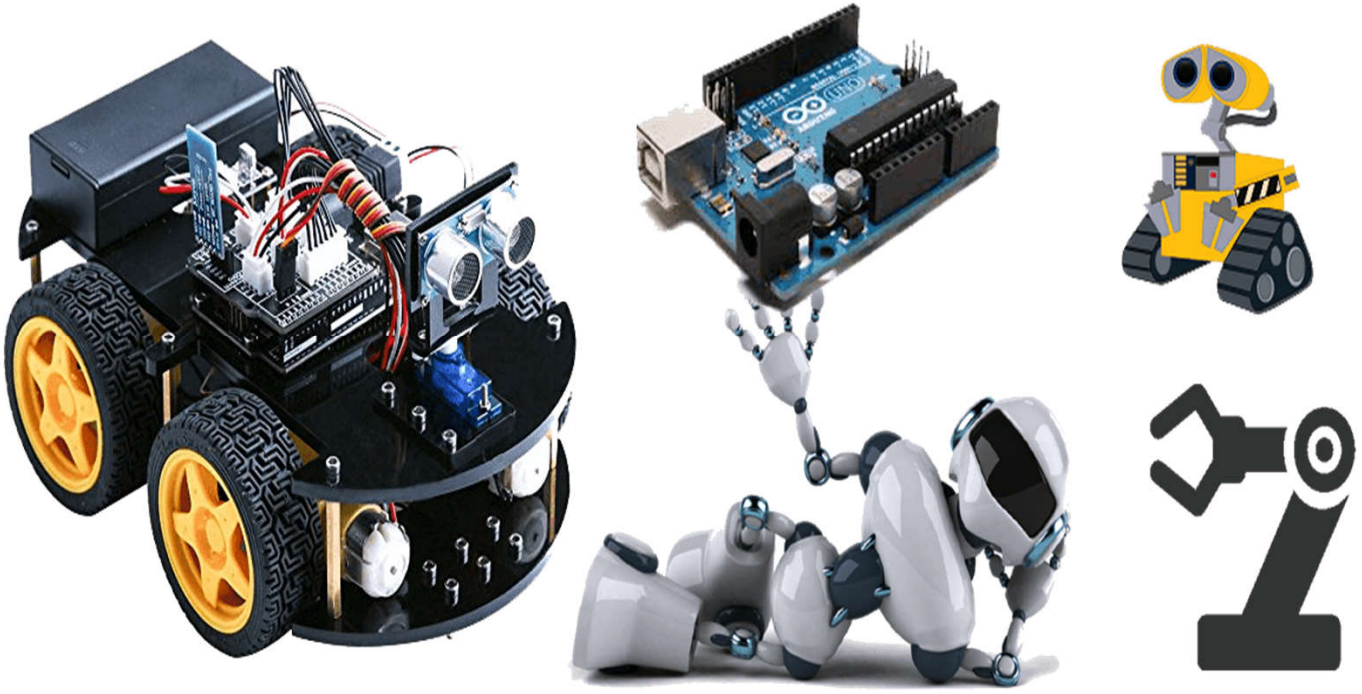
Applications of Embedded System

- Consumer Electronics



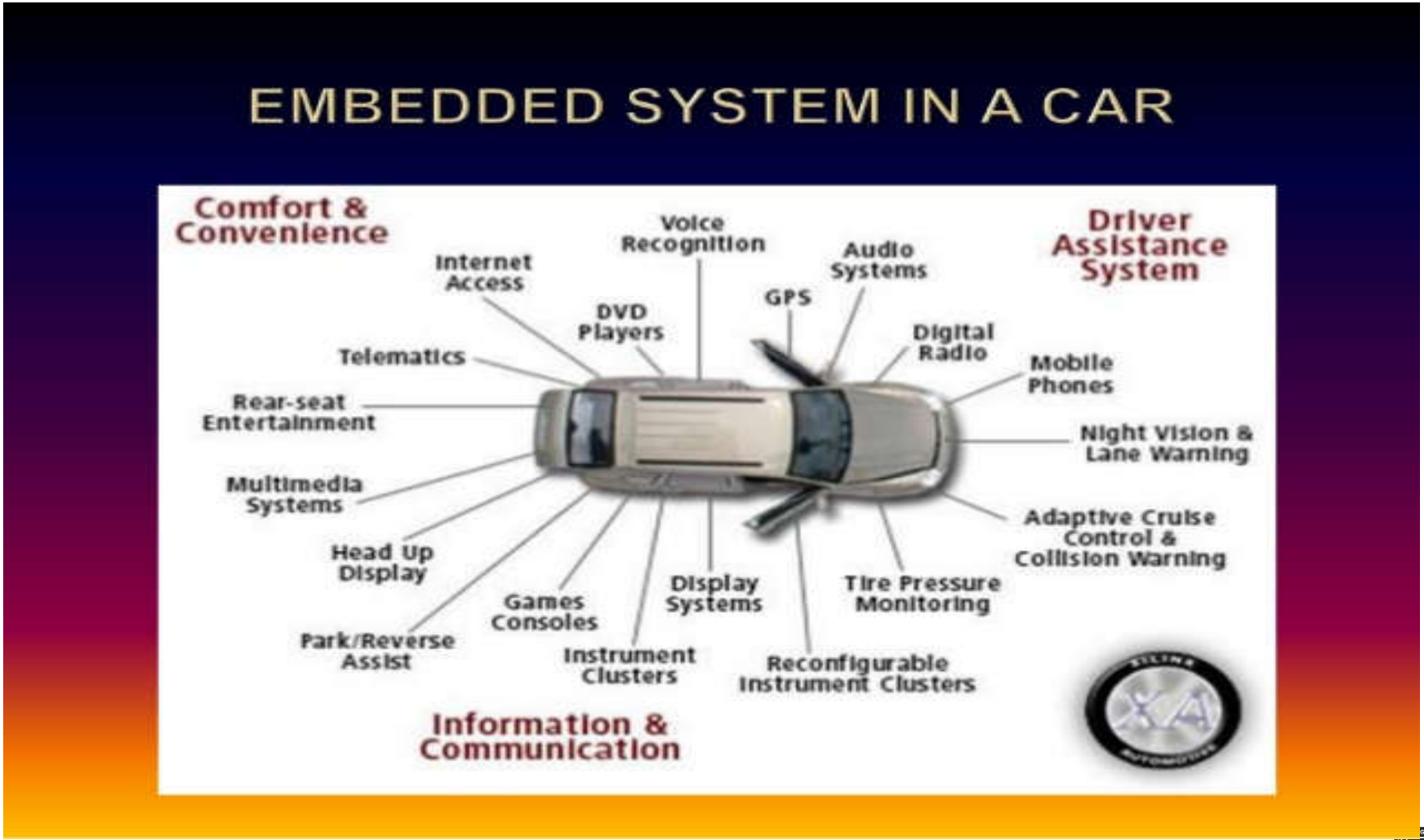
Applications of Embedded System

- Robotics



Applications of Embedded System

- Automotive Industries



Applications of Embedded System

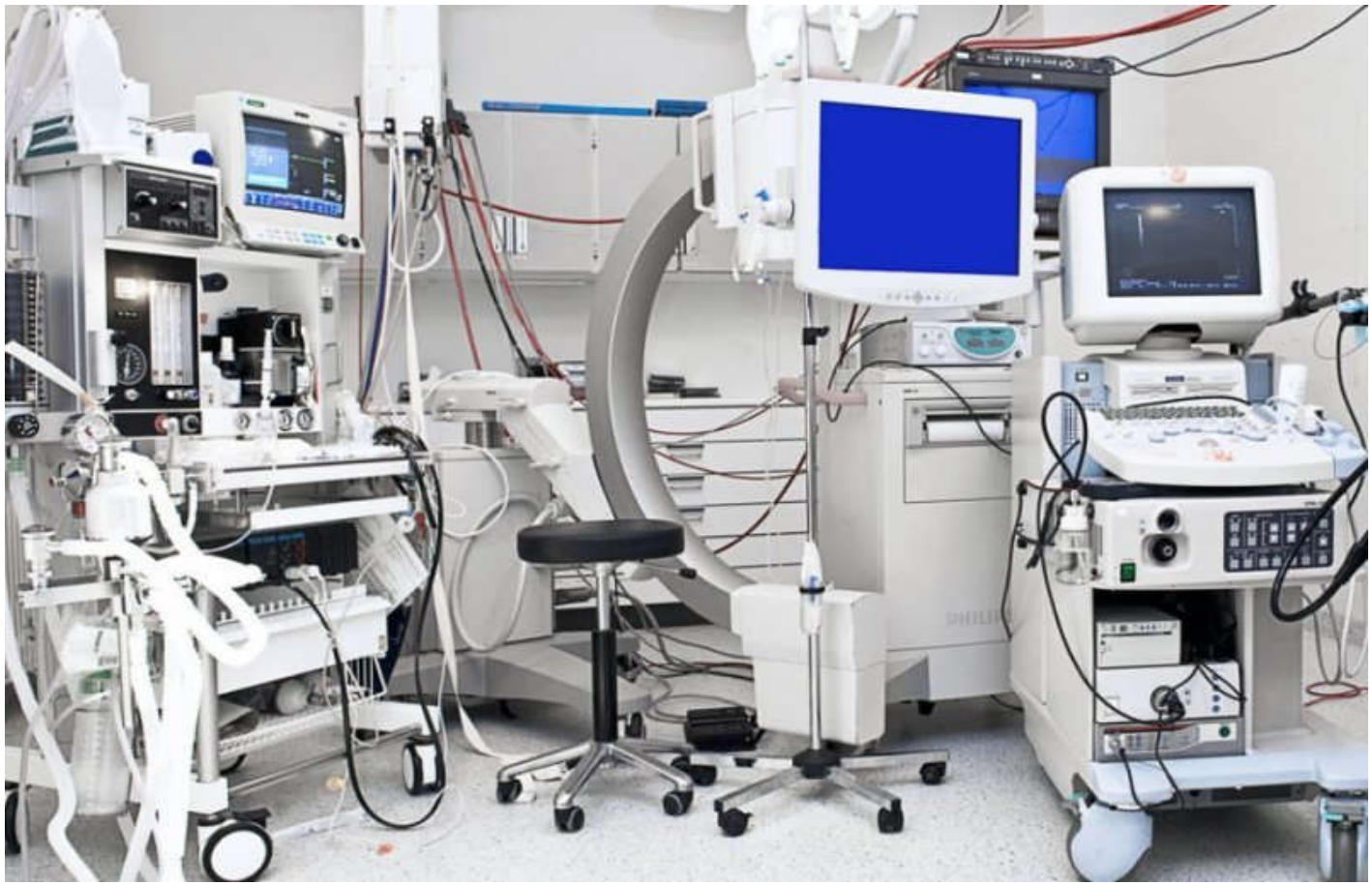
- Industrial Machines

Applications of Embedded System

- Test Equipment

Applications of Embedded System

- Medical Equipment



Applications of Embedded System

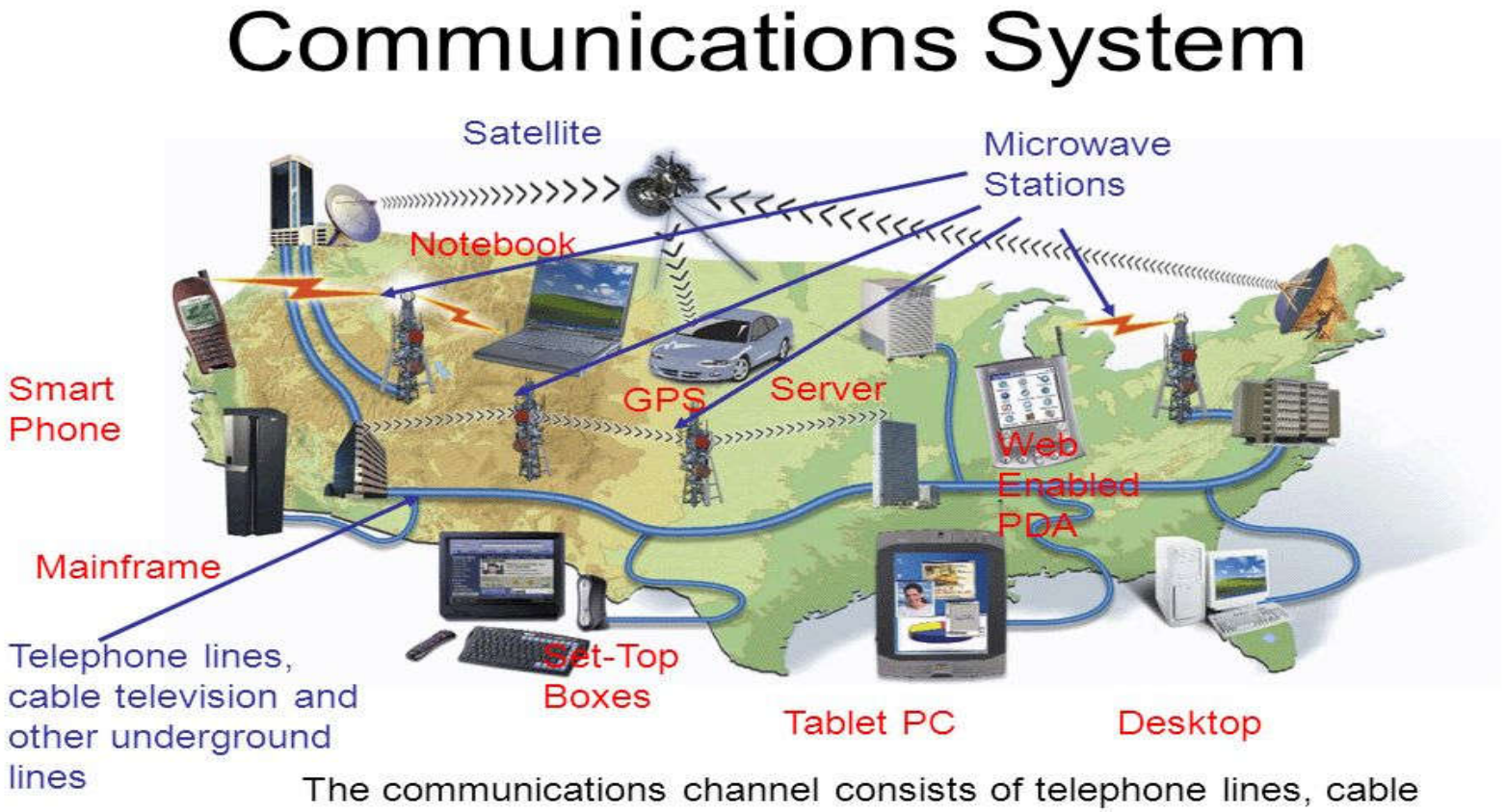
- Military Applications

Applications of Embedded System

- Aerospace

Applications of Embedded System

- Communication Systems



Applications of Embedded System

- Navigation Systems

Advantages of Embedded System

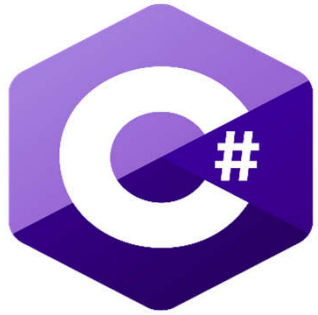
- Convenient for mass production to lower the price per piece
- Highly stable and reliable
- Small in size
- Systems are fast and less power
- Improves product quality
- Makes design smarter and better connectivity
- Enables Communications Protocols
- Flexibility of Design
- Ease of Maintenance

Disadvantages of Embedded System

- Difficult to program, need to study programming languages
- Difficult to debug and troubleshoot, need a debugger, logic/protocol analyzer, reliable scope
- Electrostatic Discharge Problem

Disadvantages of Embedded System

- Difficult to program, need to study programming languages
  - Assembly
  - C/C++
  -BASIC
  - Python
  - Java/C#
  - FORTH

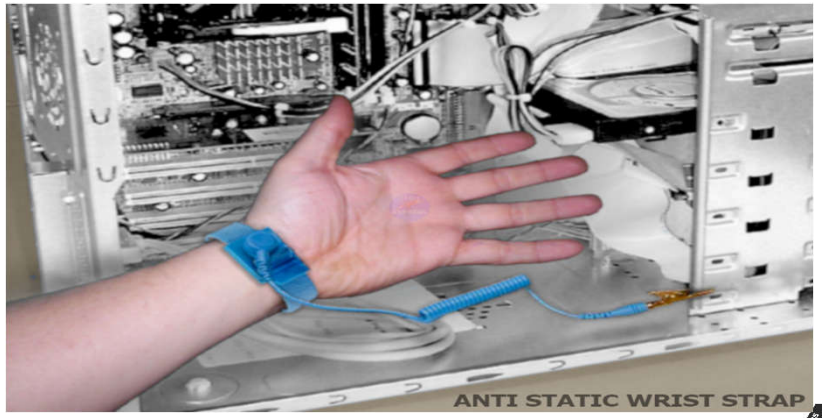


Disadvantages of Embedded System

- Difficult to debug and troubleshoot, need a debugger, logic/protocol analyzer, reliable scope

Disadvantages of Embedded System

- Electrostatic Discharge Problem



Types of Embedded System

Based on Hardware/Controller

1. Microprocessor
2. Microcontroller
3. System on Chip
4. FPGA
5. ASICS

Based on Functions

1. Real Time-provides output within a defined specific time.
   -Soft Real Time
   -Hardware Real Time
2. Stand-Alone-self-sufficient and do not depend on a host system.
3. Networked-depend on connected network to perform its assigned tasks.

Types of Embedded System

Based on Hardware/Controller

1. Microprocessor
2. Microcontroller

MPU and MCU Basics:

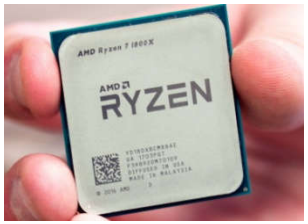
What is a Microprocessor Unit (MPU)?

What is a Microcontroller Unit (MCU)?

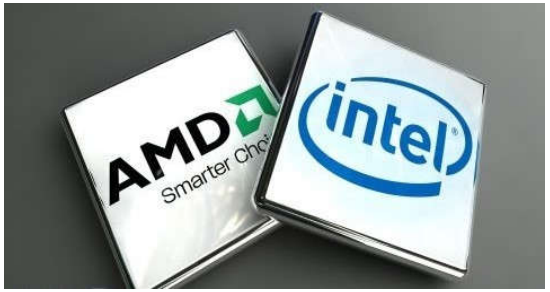
Types of Embedded System

Based on Hardware/Controller

What is a Microprocessor Unit (MPU)?



VS

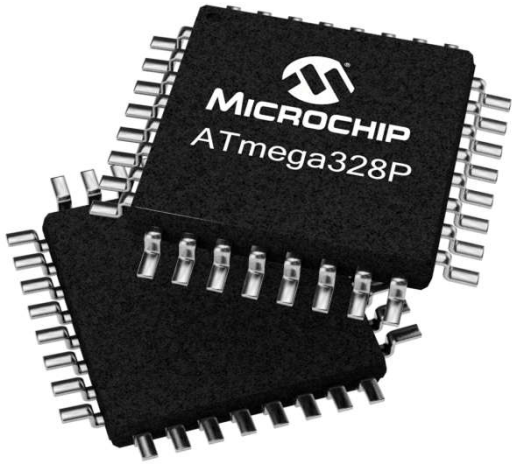


- Central Processing Unit (CPU)
- General purpose, programmable device
- Accepts digital data, processes it, provides results

Types of Embedded System

Based on Hardware/Controller

What is a Microcontroller Unit (MCU)?

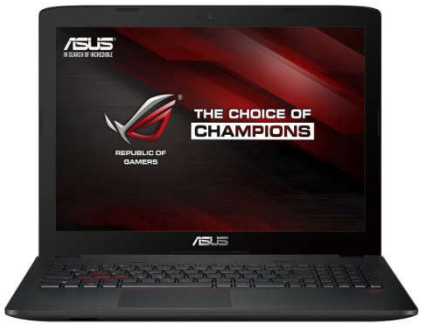


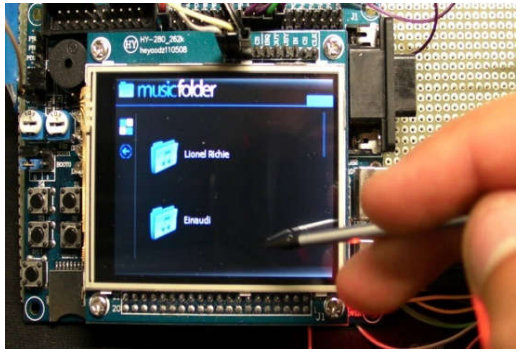
- Microprocessor + memory + I/O devices
- Basically a tiny computer on a single chip

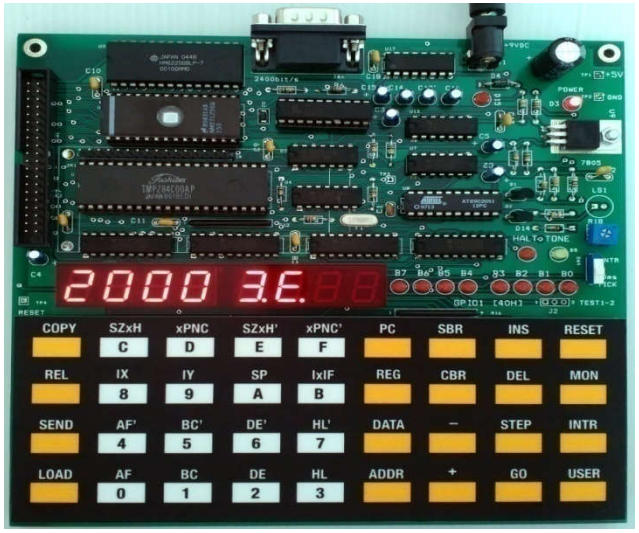
Types of Embedded System

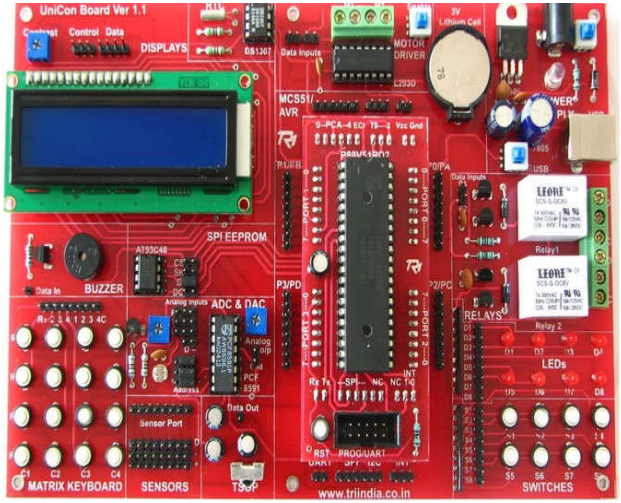
Based on Hardware/Controller

MPU vs MCU









Types of Embedded System

Based on Hardware/Controller

What is a Microcontroller Unit (MCU)?

- **small computer on a single integrated circuit.**
- **for embedded applications,**
- **microprocessors used in personal computers**
- **one or more CPUs with memory and programmable GPIOs. Program memory -Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip**
- **small amount of RAM.**
- **automatically controlled products and devices- automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems.**

Types of Embedded System

Based on Hardware/Controller

Features of a Microcontroller Unit (MCU)

- **central processing unit**
- **8-bit ,16-bit to complex 32-bit or 64-bit processors**
- **volatile memory (RAM) for data storage**
- **ROM, EPROM, EEPROM or Flash memory for program and operating parameter storage**
- **GPIOs**
- **serial input/output such as serial ports (UARTs)**
- **serial communications interfaces like I²C, Serial Peripheral Interface and Controller Area Network for system interconnect**
- **peripherals such as timers, event counters, PWM generators, and watchdog**
- **clock generator - often an oscillator for a quartz timing crystal, resonator or RC circuit**
- **analog-to-digital converters, or digital-to-analog converters**
- **in-circuit programming and in-circuit debugging support**

Types of Embedded System

Based on Hardware/Controller

Kinds of Microcontroller Unit (MCU)

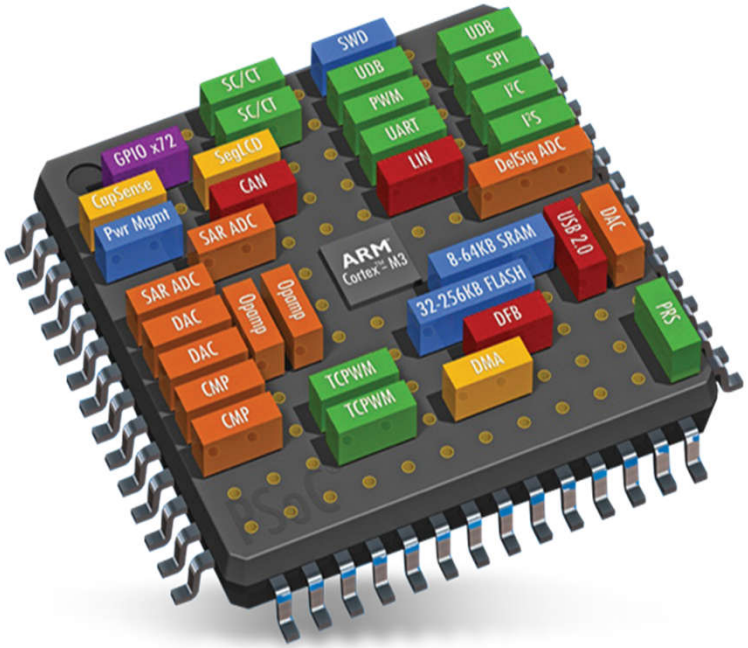
- Microchip-PICMicro-PIC12,PIC16,PIC18,PIC24,PIC32,dsPIC
- **Atmel- SAM, Atmega, Attiny**
- ARM – Texas Instruments, STMicro, Silabs, Infineon, etc
- ZILOG
- Cypress PSoC
- PLC-using realtime MCUs
- TI- MSP430
- **Tensilica-ESP8266(NodeMCU)**

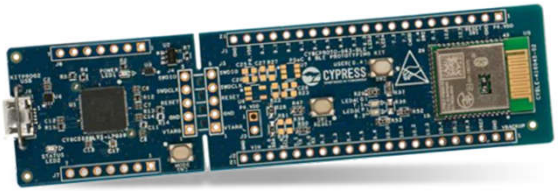
Types of Embedded System

Based on Hardware/Controller

2. System on Chip

-is an integrated circuit (also known as a "chip") that integrates all or most components of a computer or other electronic system.





Types of Embedded System

Based on Hardware/Controller

4. FPGA

-Field Programmable Gate Array

-"Simple' 'Programmable Logic Blocks

-Massive Fabric of Programmable Interconnects

-Standard CMOS Integrated Circuit fabrication process as for -SRAM memory chips (Moore's Law)

-"Hard blocks" for complex high speed functions

Huge Density of Logic Block 'Islands'

1,000 … 100,000's

in a 'Sea' of Interconnects

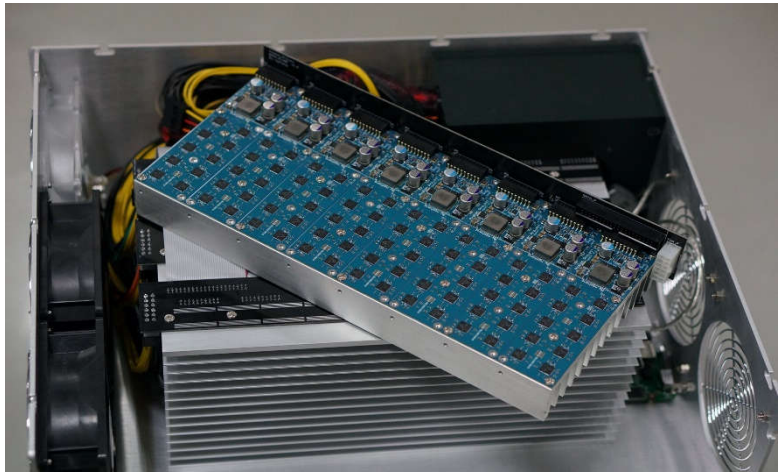
Types of Embedded System

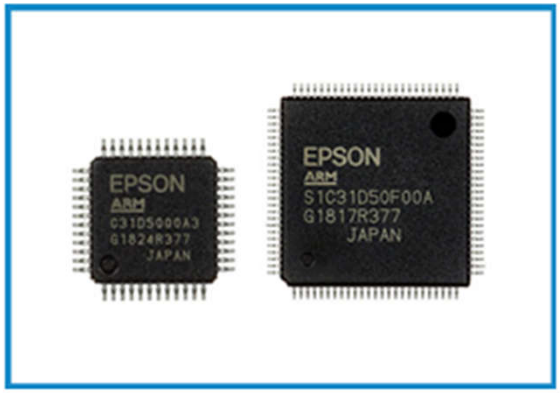
Based on Hardware/Controller

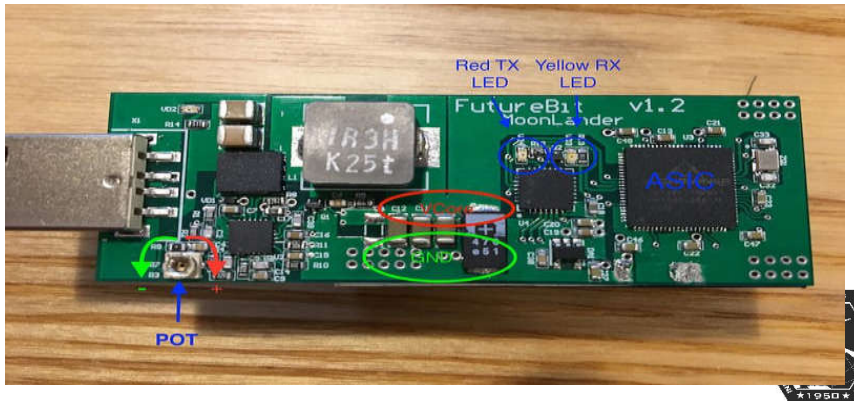
5. ASIC

Application-Specific Integrated Circuit

-integrated circuit (IC) chip customized for a particular use, rather than intended for general-purpose use.







Types of Embedded System

Based on Functions

1. Real Time-provides output within a defined specific time.

-Soft Real Time

- Soft real-time systems are typically those used where there is some issue of *concurrent access* and the need *to keep a number of connected systems up to date with changing situations*.
- Example1: The flight plans management system
  - » The software that maintains and updates the flight plans for commercial airliners.
  - » These can operate to a latency of seconds.
- Example2: Live audio-video systems
  - » Violation of constraints results in degraded quality, but the system can continue to operate.

-Hardware Real Time

- The correctness of an operation depends not only upon the logical correctness of the operation but also upon the time at which it is performed.
- Hard real-time systems are typically found interacting at a low level with physical hardware, in embedded systems.
- Example: Car Engine Control System
  - » A delayed signal may cause engine failure or damage.
- Other examples
  - » Nuclear power stations
  - » Car airbags

Types of Embedded System

Based on Functions

2. Stand-Alone-self-sufficient and do not depend on a host system.
3. Networked-depend on connected network to perform its assigned tasks.

Elements of Embedded Systems

3 Elements of Embedded Systems

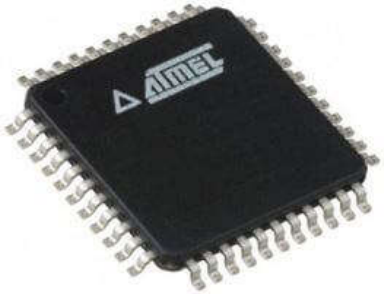
1. Hardware
2. Software
3. Firmware

Elements of Embedded Systems

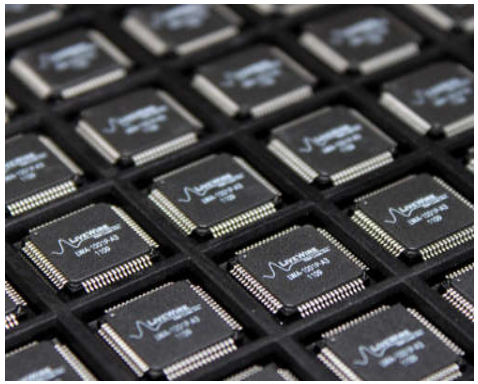
3 Elements of Embedded Systems

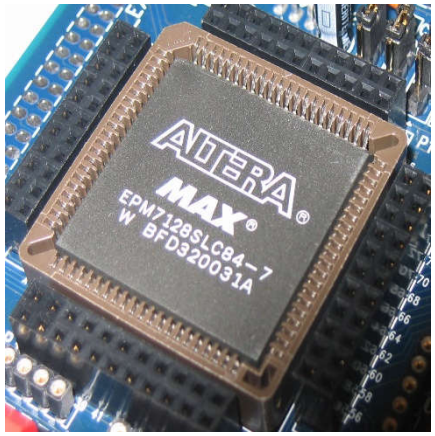
1. Hardware

VS



MPU MCU





Elements of Embedded Systems

3 Elements of Embedded Systems

2. Software

Elements of Embedded Systems

3 Elements of Embedded Systems

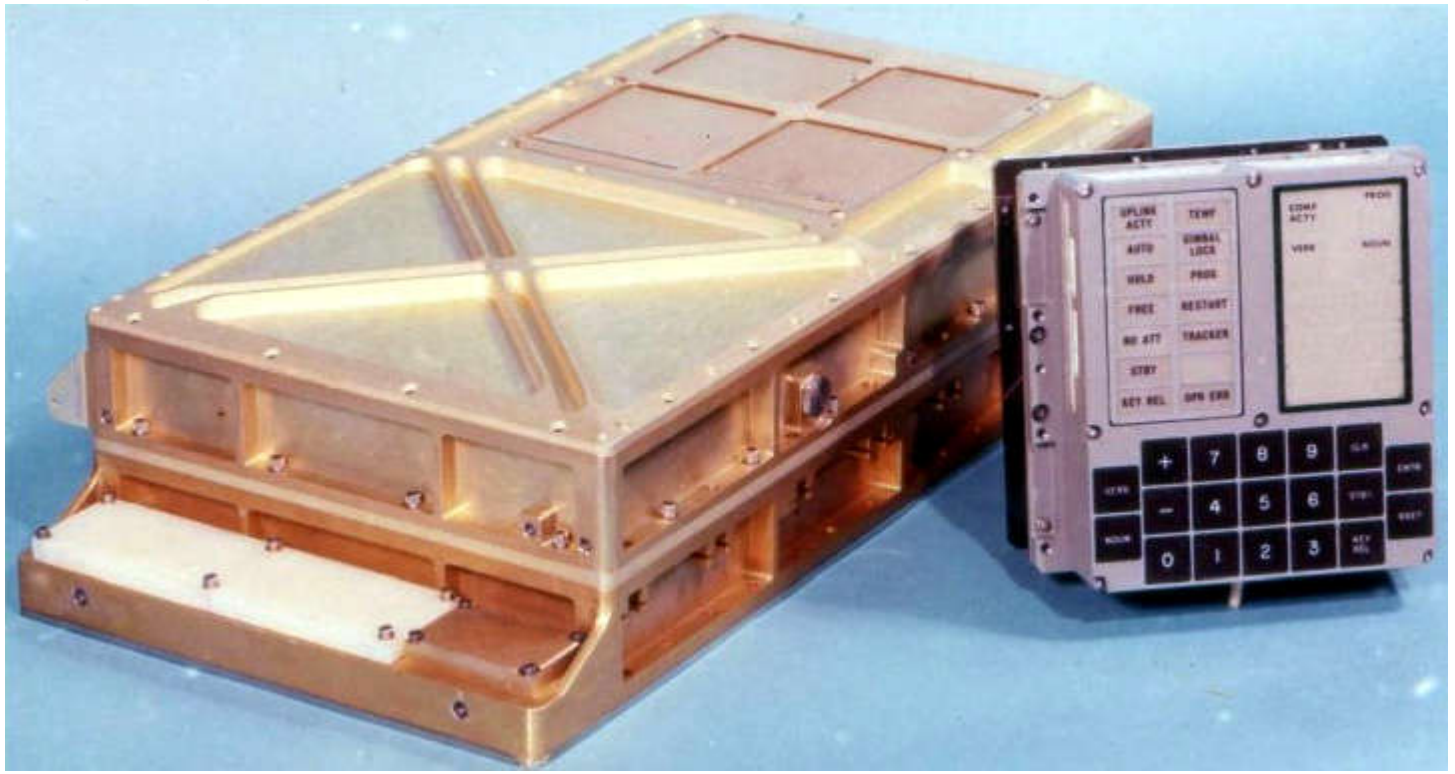
3. Firmware

```
MOVS    R2, #0x68          ; Rd = Op2
MOV     R0, R4             ; Rd = Op2
LDR     R1, =0x20003AA0    ; Load from Memory
BL      sub_8035060        ; Branch with Link
ADD     R0, SP, #0x330+var_128 ; Rd = Op1 + Op2
MOVS    R2, #0x80          ; Rd = Op2
LDR     R1, =AES_Key       ;
BL      Ref_sbox_1         ; Branch with Link
ADD     R0, SP, #0x330+var_128 ; Rd = Op1 + Op2
MOVS    R1, #1             ; Rd = Op2
MOVS    R2, #0x70          ; Rd = Op2
LDR     R3, =AES_IV        ; Load from Memory
STMEA.W SP, {R4,R5}        ; Store Block to Memory
BL      Ref_Ref_sbox       ; Branch with Link
MOVS    R0, #0x70          ; Rd = Op2
ADD.W   SP, SP, #0x324     ; Rd = Op1 + Op2
POP     {R4,R5,PC}         ; Pop registers
; End of function AES
```

```
uint64_t nLastBlockTx = 0;
uint64_t nLastBlockWeight = 0;

int64_t UpdateTime(CBlockHeader* pblock, const Consensus::Params& consensusParams,
{
    int64_t nOldTime = pblock->nTime;
    int64_t nNewTime = std::max(pindexPrev->GetMedianTimePast()+1, GetAdjustedTime

    if (nOldTime < nNewTime)
    pblock->nTime = nNewTime;

    // Updating time can change work required on testnet:
    if (consensusParams.fPowAllowMinDifficultyBlocks)
    pblock->nBits = GetNextWorkRequired(pindexPrev, pblock, consensusParams);

    return nNewTime - nOldTime;
}

BlockAssembler::Options::Options() {
    blockMinFeeRate = CFeeRate(DEFAULT_BLOCK_MIN_TX_FEE);
    DEFAULT_BLOCK_MAX_WEIGHT;
```

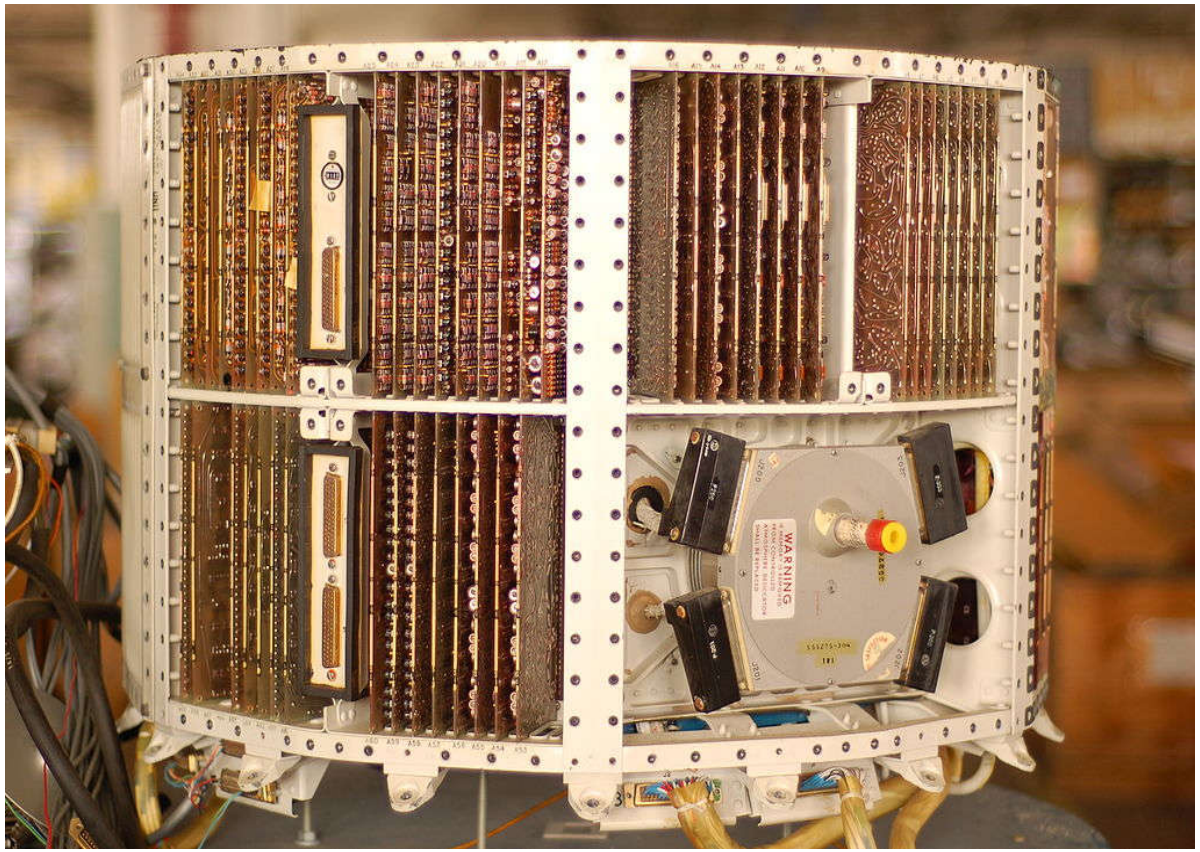
History of Embedded System

1961- Charles Stark Draper developed an integrated circuit (IC) to reduce the size and weight of the Apollo Guidance Computer, the digital system installed on the Apollo Command Module and Lunar



History of Embedded System

1965- Autonetics, now a part of Boeing, developed the D-17B, the computer used in the Minuteman I missile guidance system.



History of Embedded System

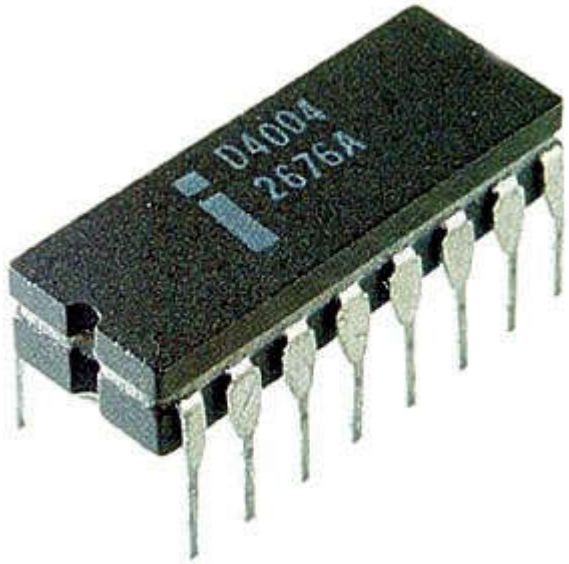
1968- First embedded system for a vehicle was released; the Volkswagen 1600 used a microprocessor to control its electronic fuel injection system.

History of Embedded System

1971- First microcontroller was developed by Texas Instruments, the TMS 1000 series, which became commercially available in 1974, contained a 4-bit processor, read-only memory (ROM) and random-access memory (RAM), and cost around $2 apiece in bulk orders.

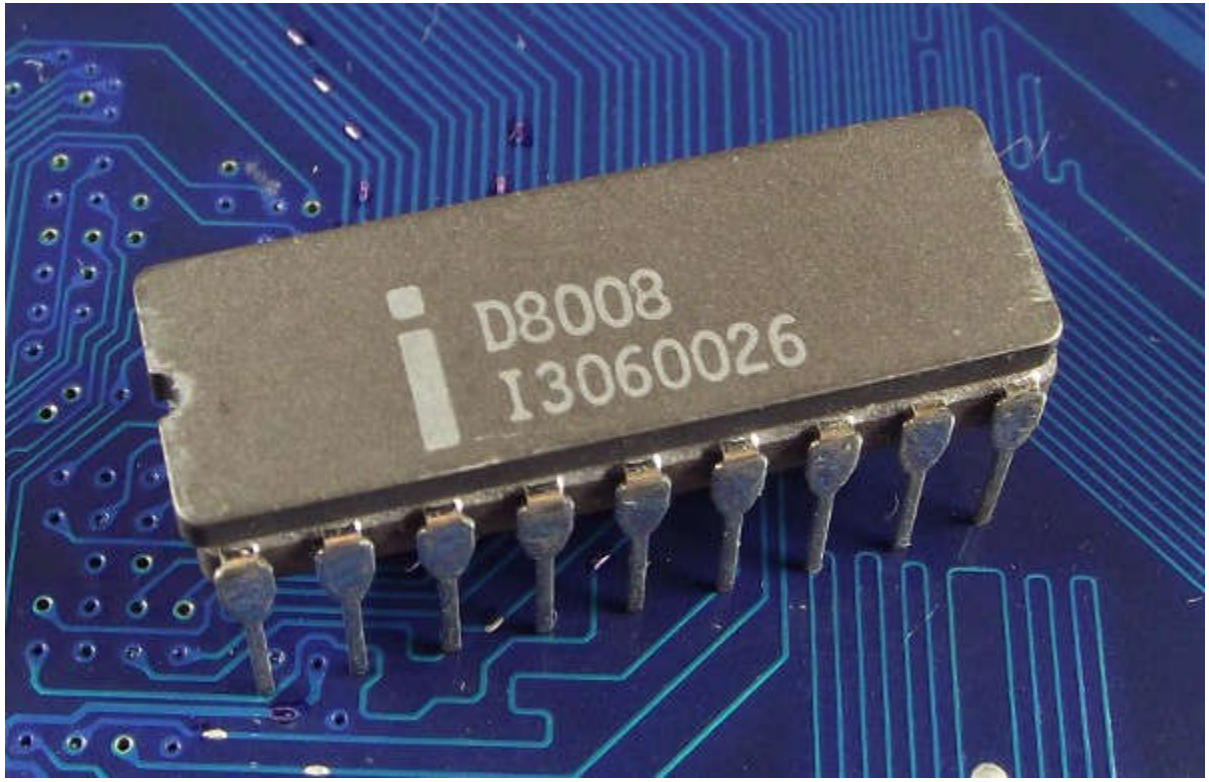
History of Embedded System

1971- Intel released what is widely recognized as the first commercially available microprocessor, the 4004. The 4-bit microprocessor was designed for use in calculators and small electronics, though it required eternal memory and support chips.



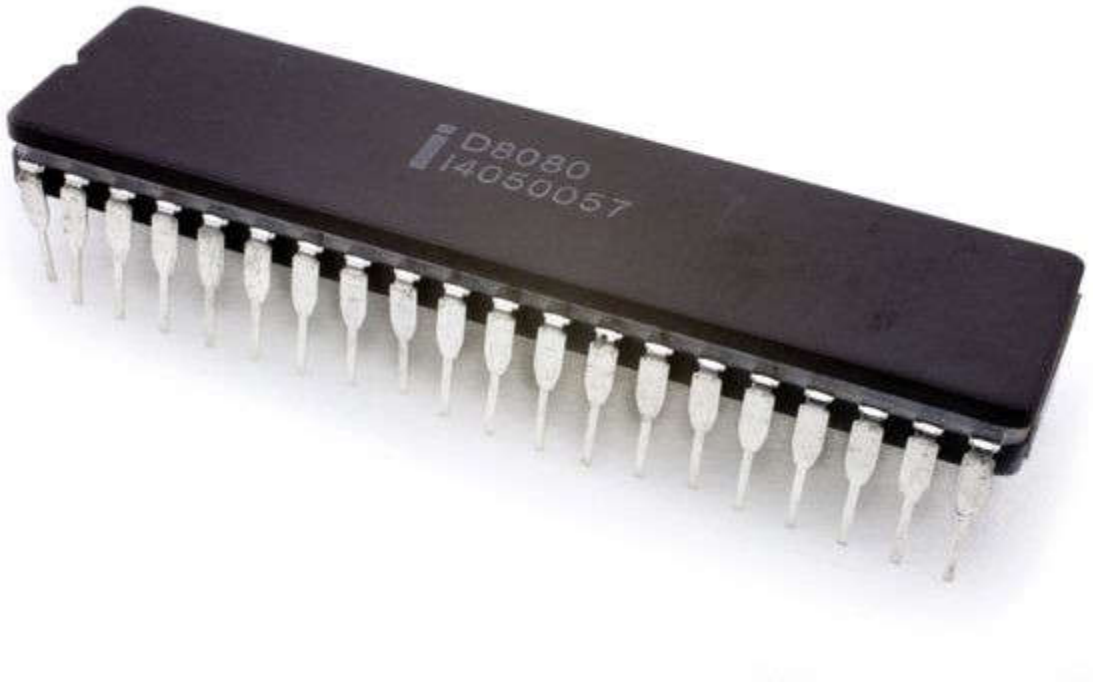
History of Embedded System

1972- The 8-bit Intel 8008, released in 1972, had 16 KB of memory.



History of Embedded System

1974- Intel 8080 followed in 1974 with 64 KB of memory.



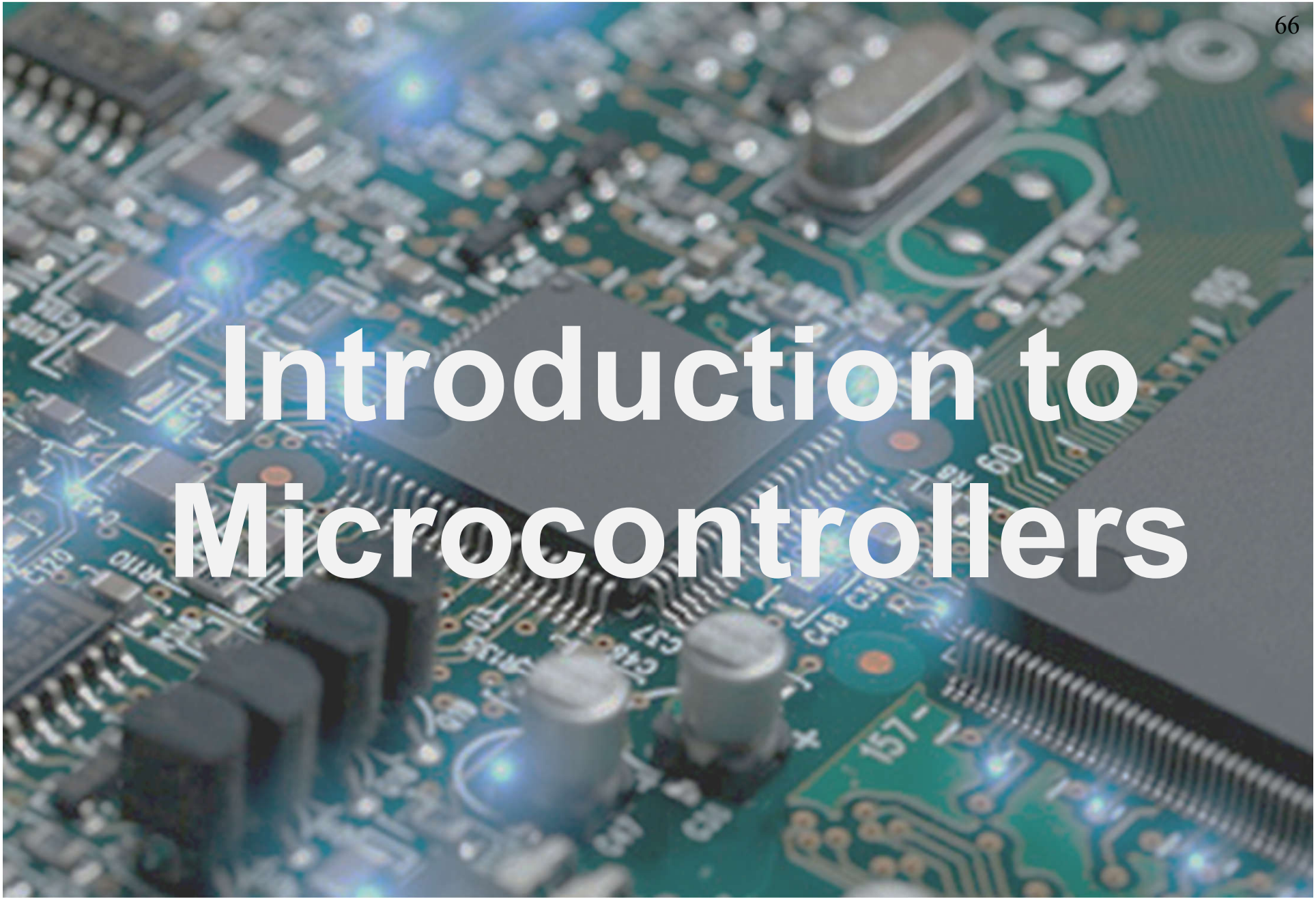
History of Embedded System

1987- First embedded operating system, the real-time VxWorks, was released by Wind River.

```
## Starting application at 0x4010100000 ...

                                        VxWorks 7 SMP 64-bit
                                  Core Kernel version: 1.0.0.0
                                  Build date: May 30 2014 10:51:05

                                  Copyright Wind River Systems, Inc.
                                            1984-2014

            Board: Wind River Dev Kit MP8
        CPU Count: 8
  OS Memory Size: 1899MB
ED&R Policy Mode: Deployed

Adding 5290 symbols for standalone.

[vxWorks]# i
  NAME          TID         PRI   STATUS        PC          ERRNO   CPU #
---------- ---------------- --- ---------- ---------------- ------- -----
tJobTask     40104cdbc0        0 PEND       401020c83c             0     -
tExcTask     40102a073c        0 PEND       401020c83c             0     -
tLogTask     40104d01d8        0 PEND       401020b0f0             0     -
tShell0      40105c1d30        1 READY      4010215e08             0     0
ipcom_tick>  401057a990       20 PEND       401020c83c             0     -
tVxdbgTask   401057dc20       25 PEND       401020c83c             0     -
tNet0        40104d3b78       50 PEND       401020c2b4             0     -
ipcom_sysl>  40104c9810       50 PEND       401020d3d4             0     -
tNetConf     40105a6e40       50 PEND       401020c83c             0     -
miiBusMoni>  40104d5e08      252 DELAY      4010215640             0     -
ipcom_egd    4010583c20      255 DELAY      4010215640             0     -
tIdleTask0   40102a2fb0      287 READY      401020c004             0     -
tIdleTask1   40102a7220      287 READY      401020c00c             0     1
tIdleTask2   40102ab490      287 READY      401020c004             0     2
tIdleTask3   40102afb20      287 READY      401020c004             0     3
tIdleTask4   40102b1700      287 READY      401020c004             0     4
tIdleTask5   40102b2440      287 READY      401020c004             0     5
tIdleTask6   40102a4620      287 READY      401020c004             0     6
tIdleTask7   40102a4860      287 READY      401020c004             0     7
[vxWorks]#
```

History of Embedded System

1996- Microsoft's Windows Embedded CE was released.



Introduction to Microcontrollers

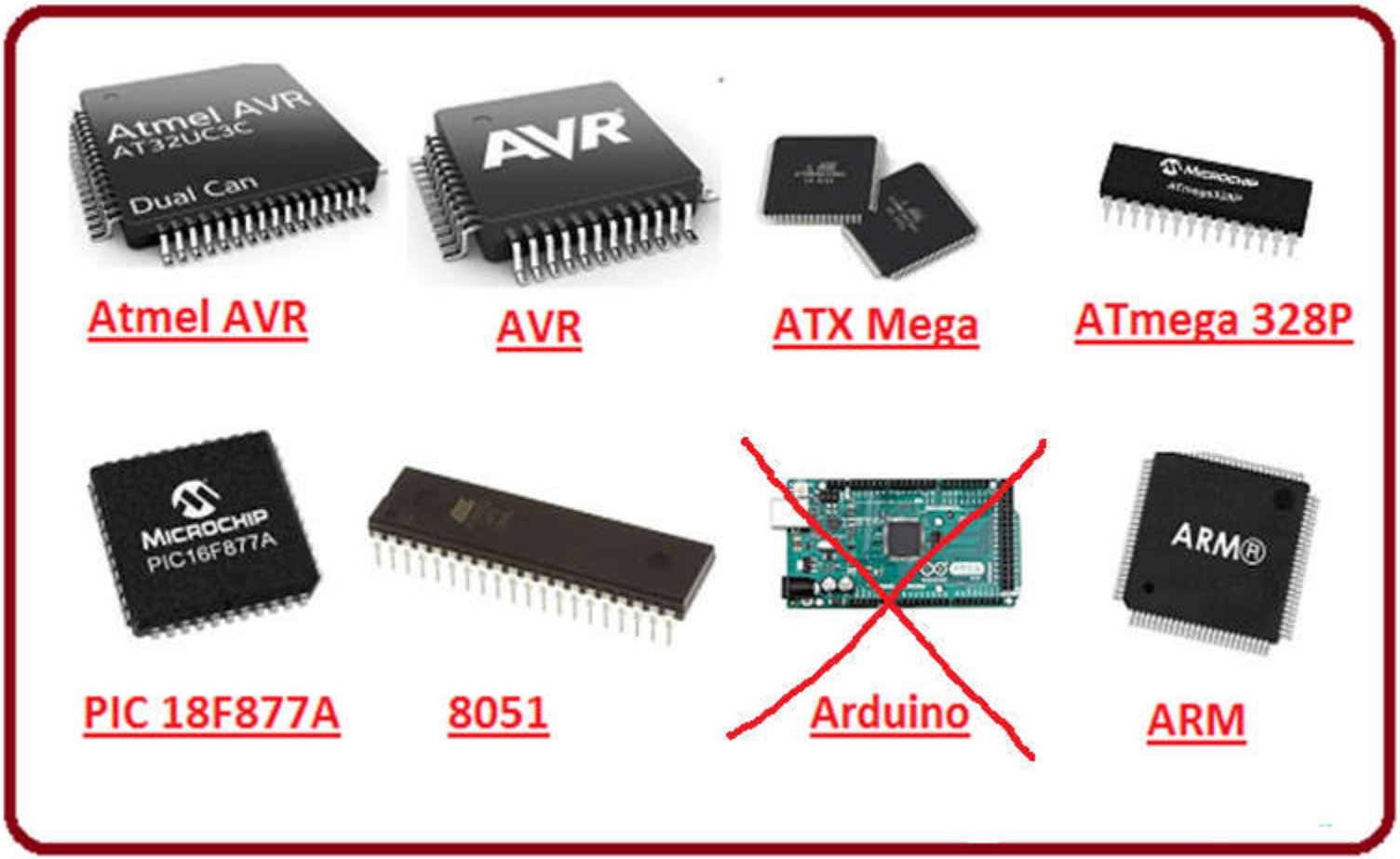
Introduction to Microcontrollers

Contents

0.1 Objectives

- In this tutorial, you will learn:
  - To understand Microcontroller basic concepts.
  - To know about the Importance of a Microcontroller.
  - To become familiar with the types of Microcontrollers.
  - To become aware of the difference between Microprocessor and Microcontroller.
  - To understand the advantages and disadvantages of a Microcontroller.
  - To appreciate the history of a Microcontroller.

0.2 Introduction



0.2 Introduction

- Every day the world is becoming more and more digital, connected, and automated.
- Hidden computers are part in our daily lives.

Examples:

Engine Control Unit in your car
Elevator in your office
Point of sale systems
Printers and scanners
Routers and switches
Machine control & monitoring devices in industrial plants

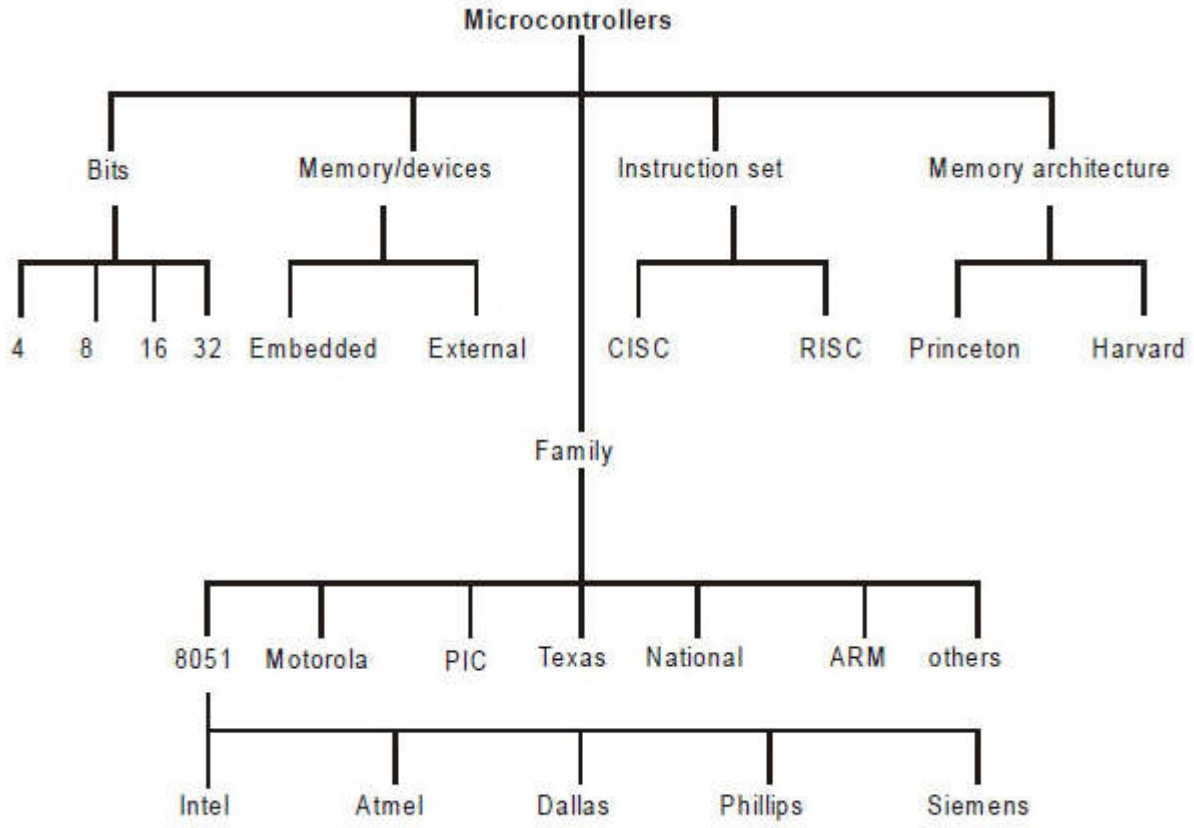
0.3 What is a Microcontroller?

- **Small Computer** on a single metal-oxide-semiconductor (MOS) integrated circuit (IC).
- Less sophisticated than System On Chip (SoC)
- Contains one or more CPUs with memory and programmable GPIOs and small RAM
- Program memory types are Ferroelectric RAM, NOR flash or OTP ROM

0.4 Why Is a Microcontroller Important?

- Miniaturization
- All in one device
- Low cost (in bulk order) and Low power*
- Bill of Materials can be reduced
- Making a device smarter
- Flexibility/Programmable
- Connectivity
- Price to Performance ratio

0.5 Types of Microcontroller



Types of microcontrollers

0.6 Other Types of Microcontroller

Digital Signal Processor (DSP)

specialized microprocessor optimized for mathematical operations in digital signal processing

Fast Fourier Transform (FFT)

Digital Filters

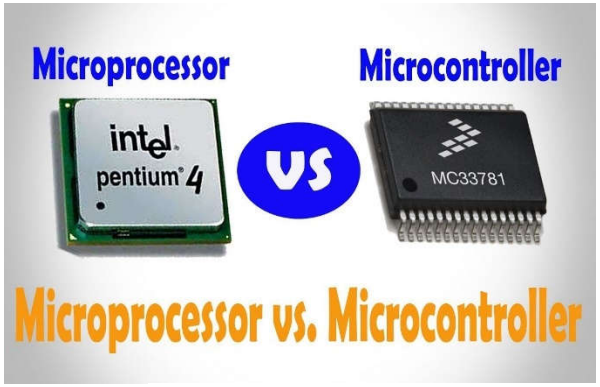
Digital Control Loops

0.6 Other Types of Microcontroller

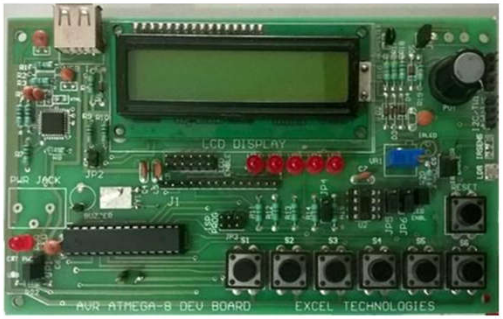
Digital Signal Controller (DSC)

DSP+memory+I/O devices

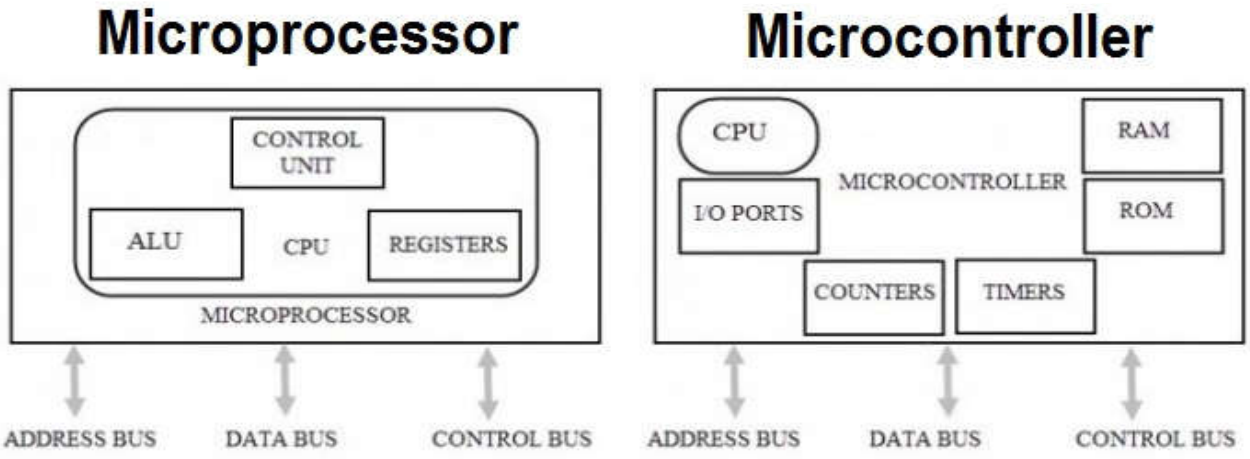
0.7 Microprocessor vs Microcontroller



MICROCONTROLLER



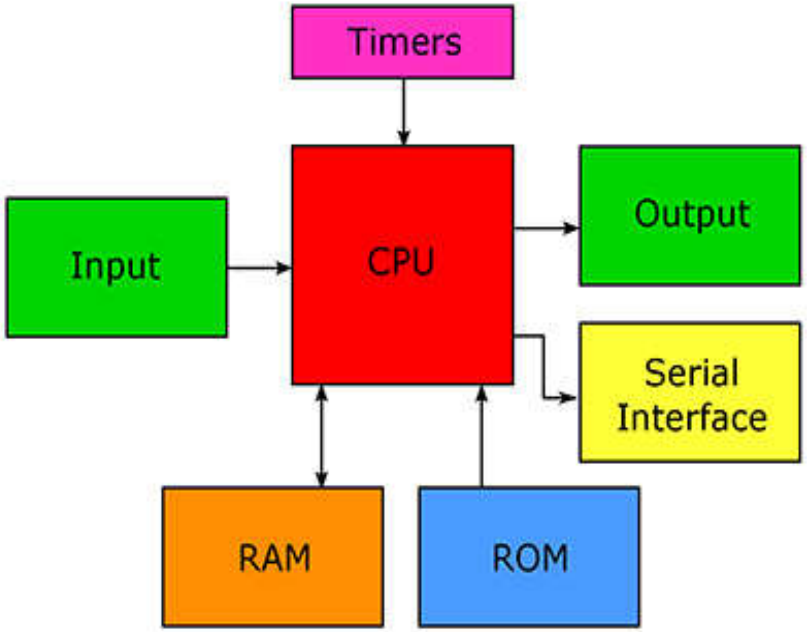
MICROPROCESSOR



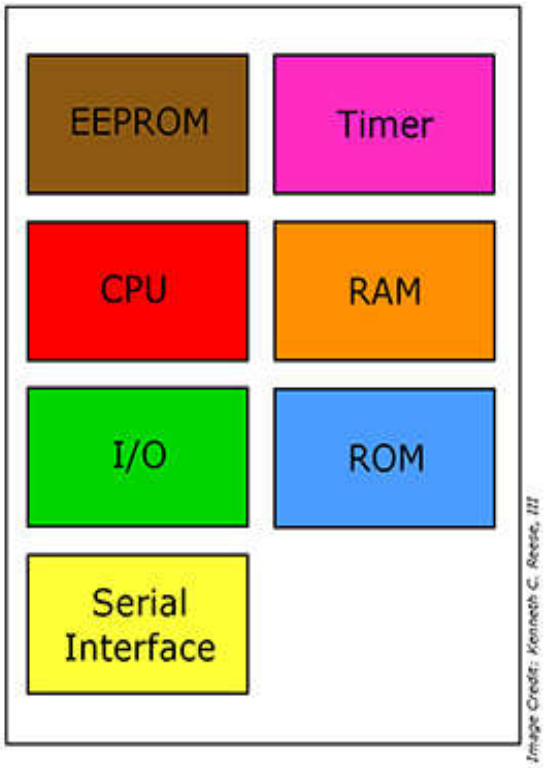
Microprocessor vs Microcontroller

0.7 Microprocessor vs Microcontroller

Microprocesser: CPU
and several supporting chips.



Microcontroller: CPU
on a single chip.



0.7 Microprocessor vs Microcontroller

Microcontroller	Difference	Microprocessor
Specific-purpose CPU runs at a lower frequency	**CPU**	General-purpose CPU runs at a higher frequency
Slower	**Speed**	Faster in various operations
All devices and peripherals are integrated inside the unit	**Components**	Components can be outside the processor
Supports	**Boolean Functions**	Doesn't support
Faster in accessing CPU registers	**CPU Registers**	Need more time to access CPU registers
Mostly in real-time applications to control a specific device	**Usage**	In desktops, laptops, servers and other general-purpose computer devices
Supports RTOS	**Real-Time Operating System**	Supports RTOS+Kernel-based services
Supports all bitwise operations	**Bitwise Operations**	Doesn't support all bitwise operations
Relatively simple	**Hardware Complexity**	More complex
Generally more affordable	**Cost**	More expensive
Doesn't always support	**Multi-tasking**	Supports
8051, PIC, MSP430, Renesas, STM microcontroller	**Example**	X86, Motorola, Broadcom, Pentium

0.8 Advantages and Disadvantages of a Microcontroller

Advantages:

Complexity is reduced vs analog/glue logic

More reliable

Enables communication protocols

Flexibility

Ease of maintenance or programmable

0.8 Advantages and Disadvantages of a Microcontroller

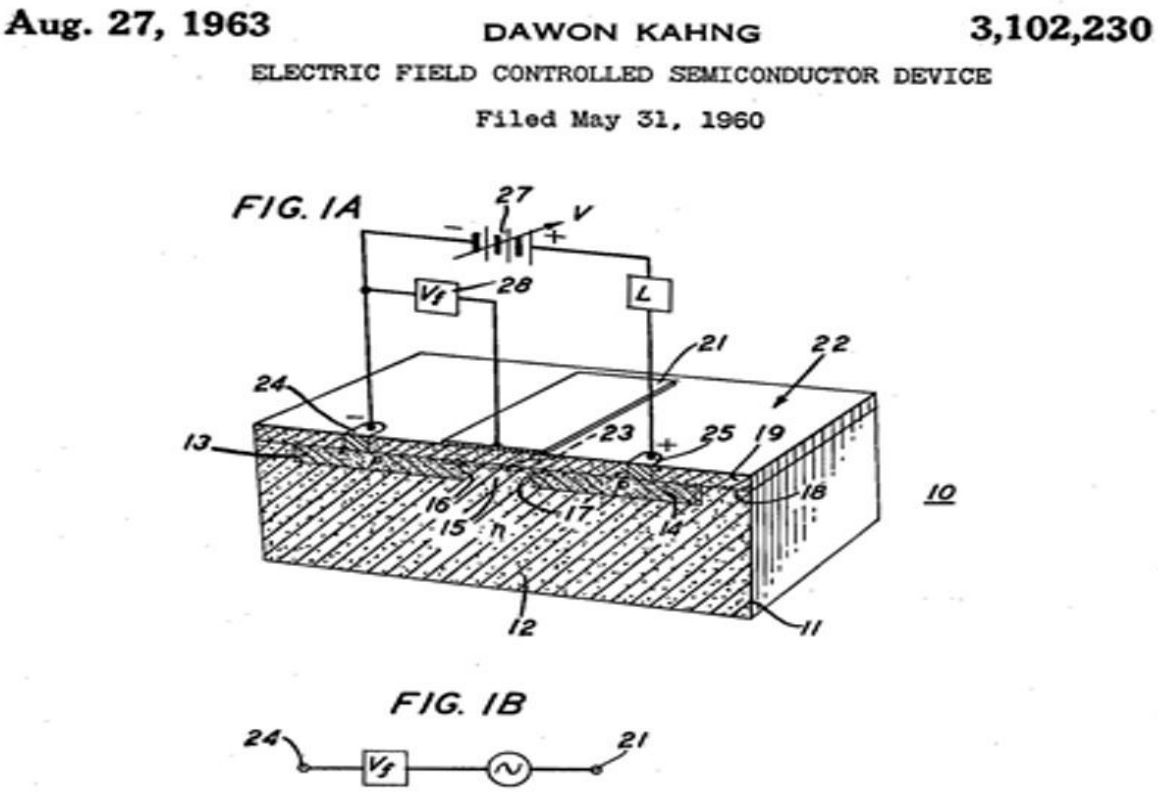
Disadvantages:

Programming complexity

Electrostatic sensitivity

0.9 The History of Microcontroller

- **1959**-Mohamed M. Atalla and Dawon Kahng at Bell labs invented the **MOSFET**



Aug. 27, 1963 DAWON KAHNG 3,102,230

ELECTRIC FIELD CONTROLLED SEMICONDUCTOR DEVICE

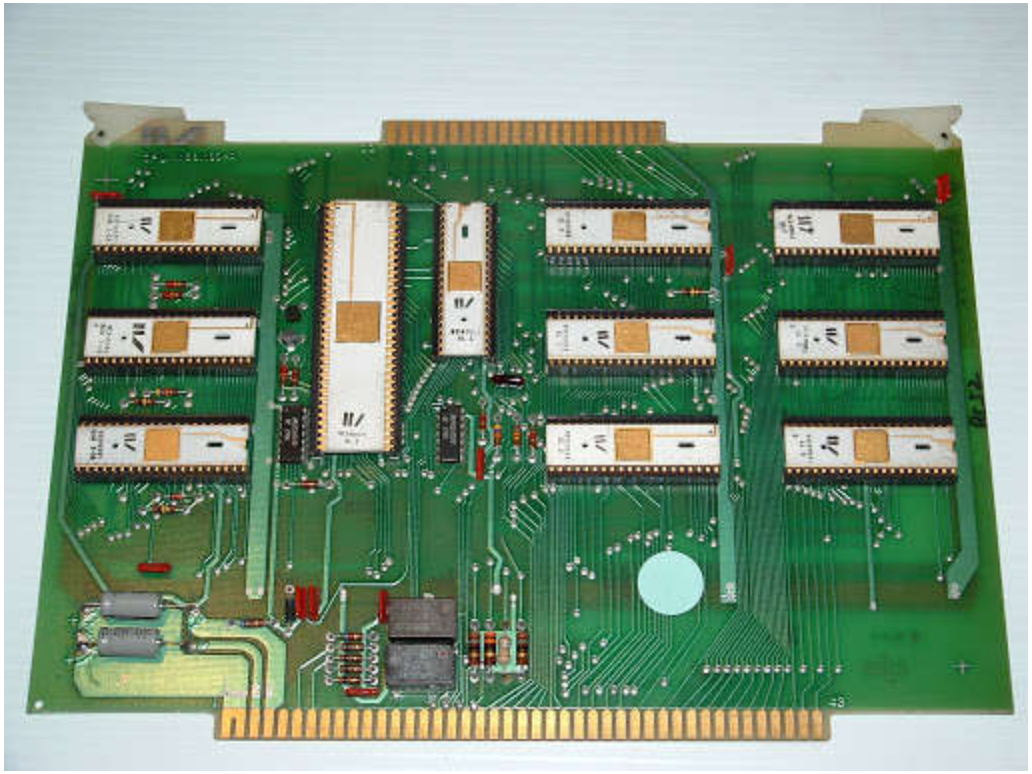
Filed May 31, 1960

FIG. 1A

FIG. 1B

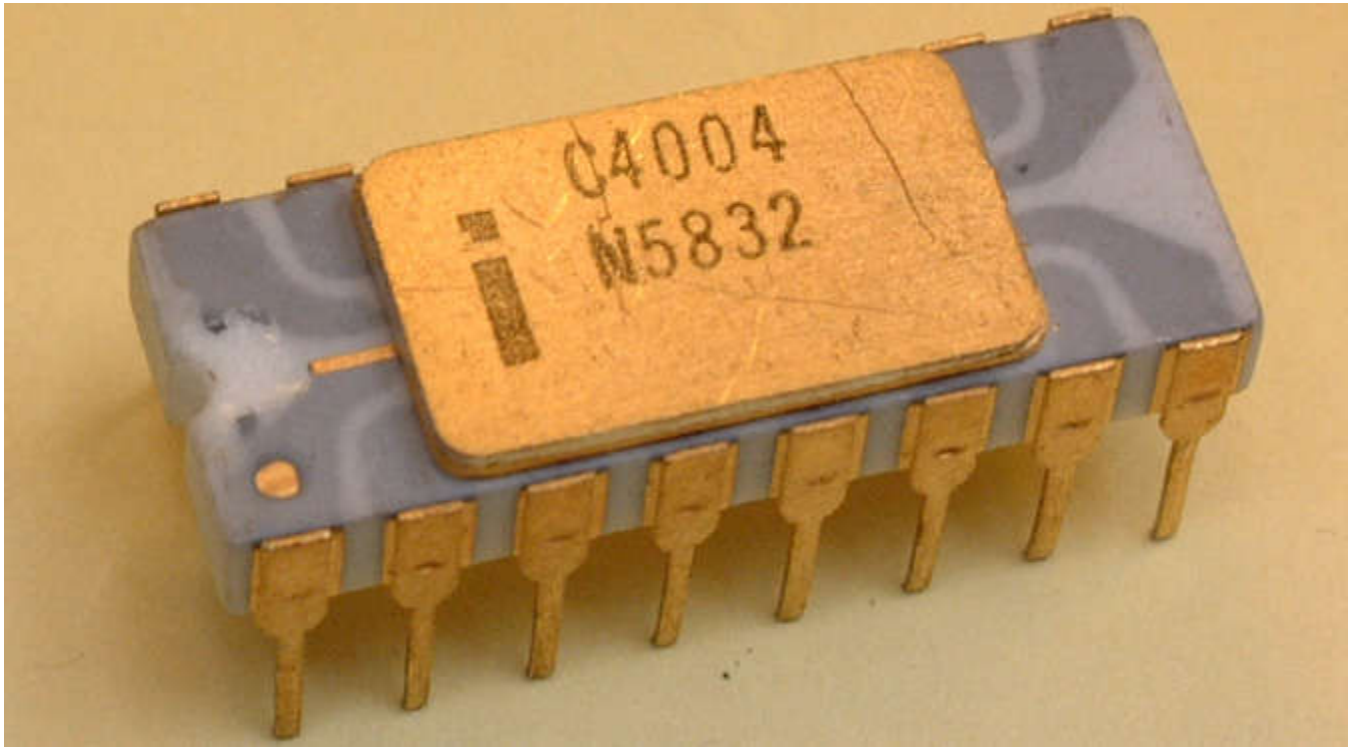
0.9 The History of Microcontroller

- Four-Phase Systems AL1 in **1969** and the Garrett AiResearch MP944 in 1970, were developed with multiple MOS LSI chips.



0.9 The History of Microcontroller

- Intel 4004, released on a single MOS LSI chip in **1971**. It was developed by Federico Faggin, using his silicon-gate MOS technology, along with Intel engineers Marcian Hoff and Stan Mazor, and Busicom engineer Masatoshi Shima.

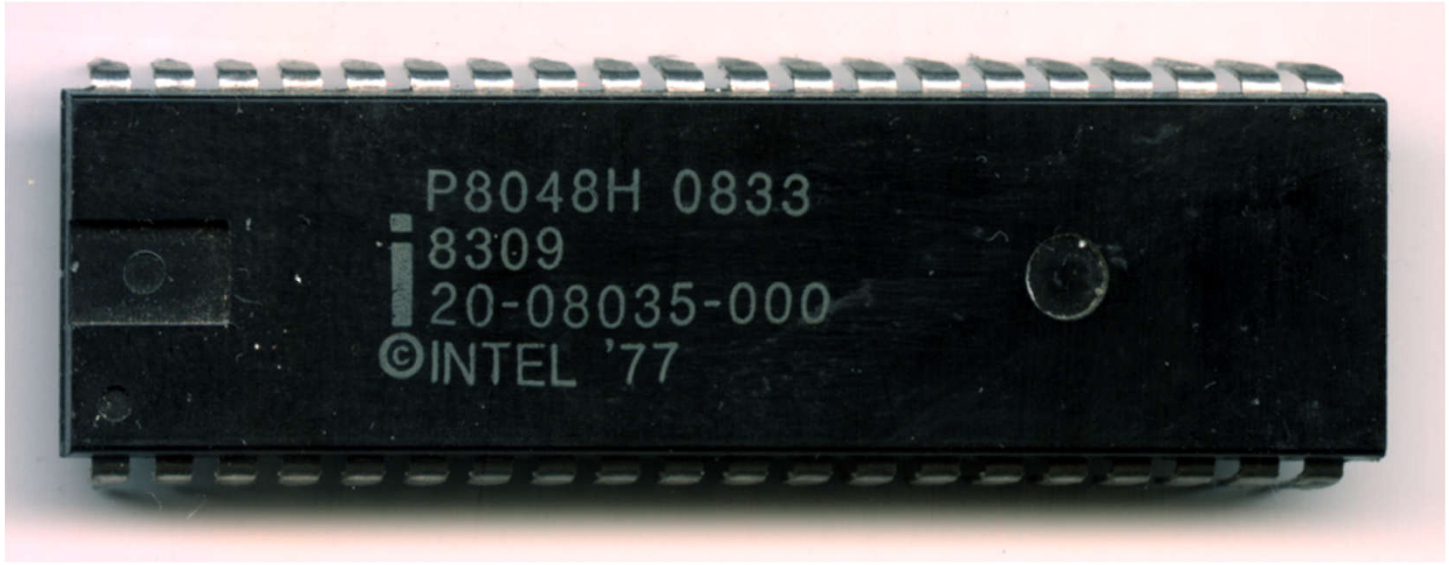


0.9 The History of Microcontroller

- TI engineers Gary Boone and Michael Cochran with the successful creation of the first microcontroller in **1971**. The result of their work was the TMS 1000, which became commercially available in 1974. It combined read-only memory, read/write memory, processor and clock on one chip and was targeted at embedded systems.

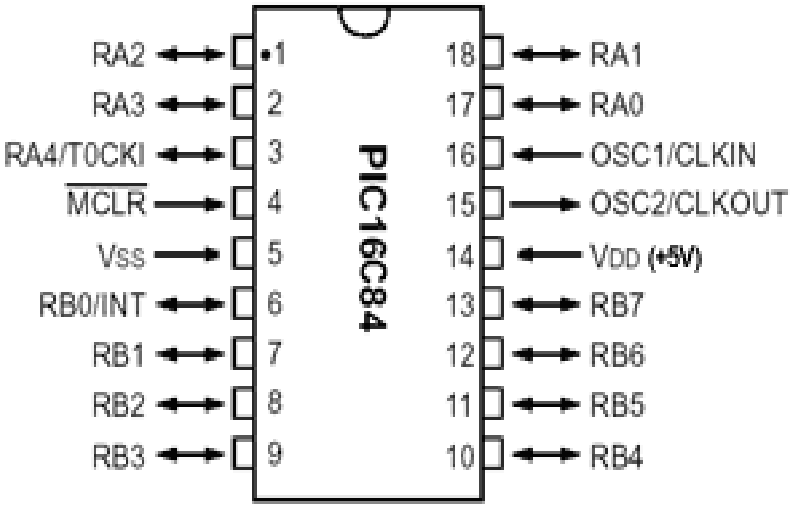
0.9 The History of Microcontroller

- Intel 8048, with commercial parts first shipping in **1977**.It combined RAM and ROM on the same chip with a microprocessor. At that time Intel's President, Luke J. Valenter, stated that the microcontroller was one of the most successful products in the company's history, and he expanded the microcontroller division's budget by over 25%.



0.9 The History of Microcontroller

- In **1993**, the introduction of EEPROM memory allowed microcontrollers (beginning with the Microchip PIC16C84)to be electrically erased quickly without an expensive package as required for EPROM, allowing both rapid prototyping, and in-system programming.



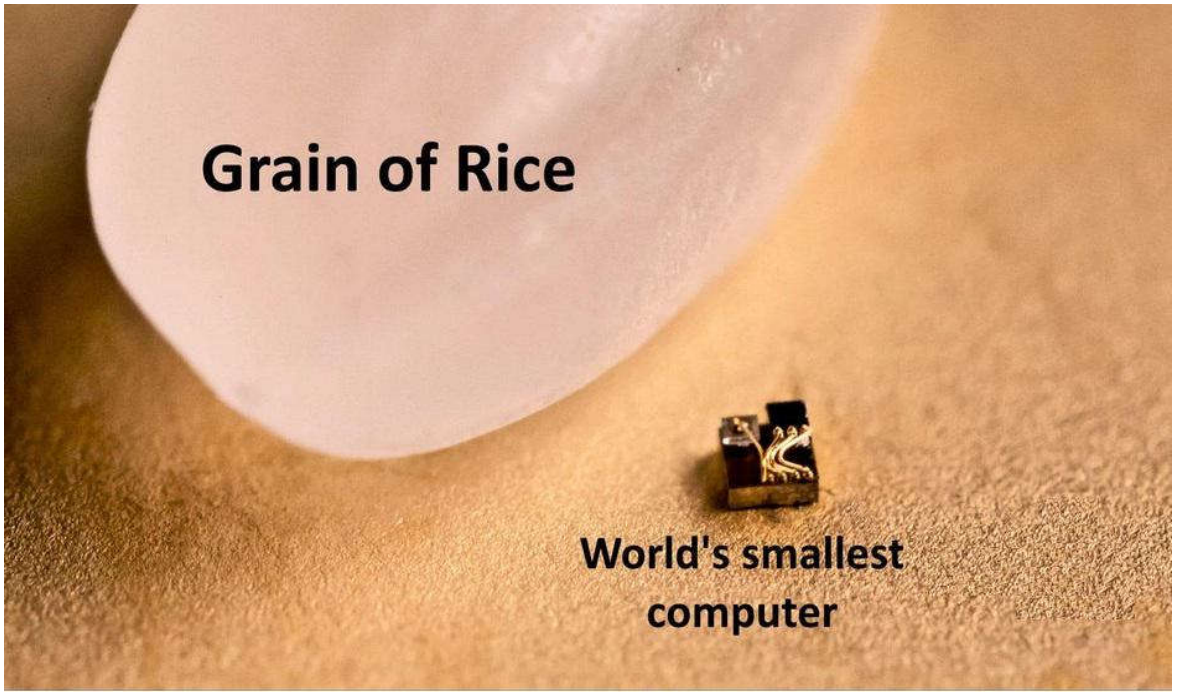
0.9 The History of Microcontroller

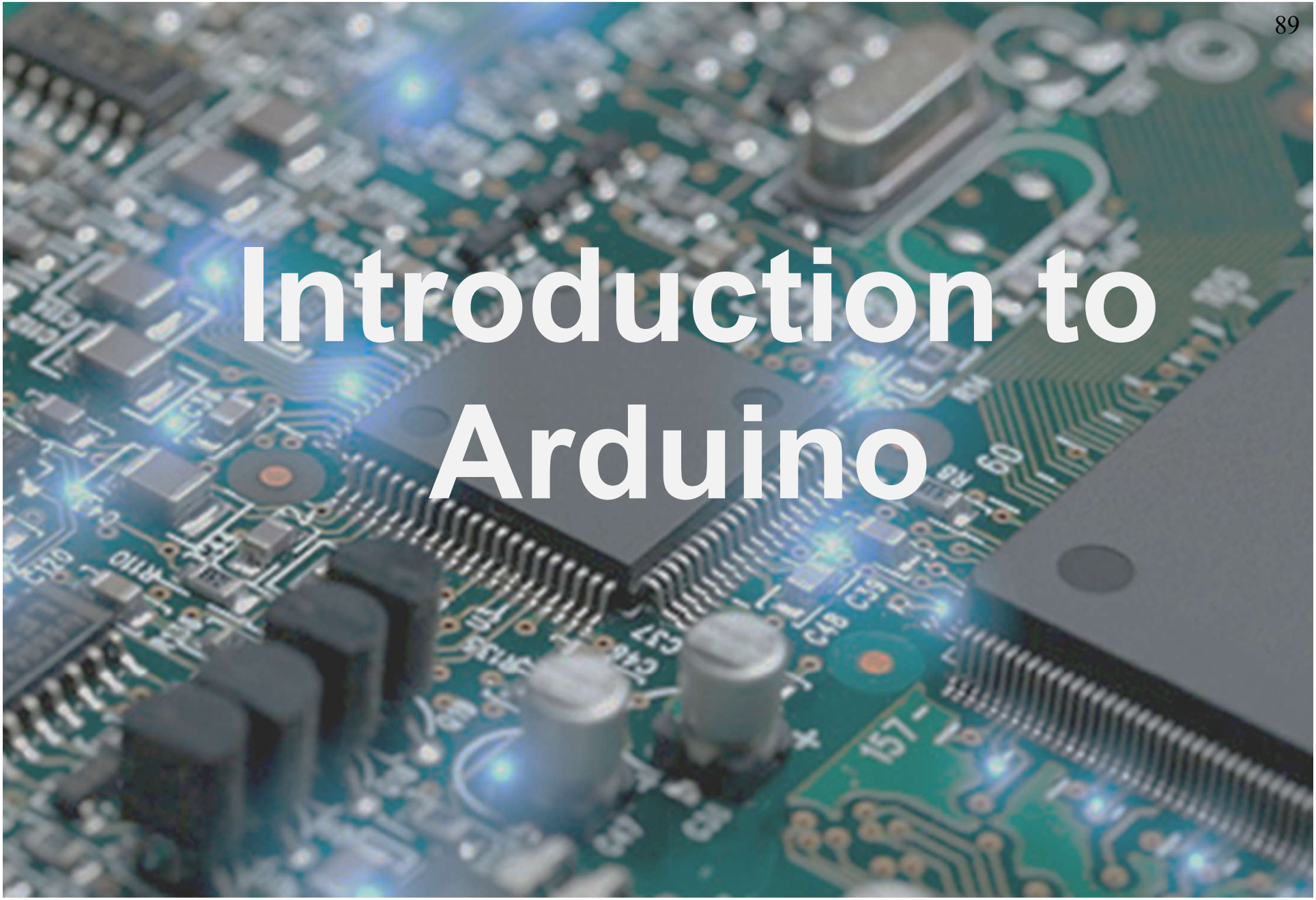
As of **2008**, there are several dozen microcontroller architectures and vendors including:

- ARM core processors (many vendors)
- ARM Cortex-M cores are specifically targeted toward microcontroller applications
- Microchip Technology Atmel AVR (8-bit), AVR32 (32-bit), and AT91SAM (32-bit)
- Cypress Semiconductor's M8C core used in their PSoC (Programmable System-on-Chip)
- Freescale ColdFire (32-bit) and S08 (8-bit)
- Freescale 68HC11 (8-bit), and others based on the Motorola 6800 family
- Intel 8051, also manufactured by NXP Semiconductors, Infineon and many others
- Infineon: 8-bit XC800, 16-bit XE166, 32-bit XMC4000 (ARM based Cortex M4F), 32-bit TriCore and, 32-bit Aurix Tricore Bit microcontrollers[34]
- Maxim Integrated MAX32600, MAX32620, MAX32625, MAX32630, MAX32650, MAX32640
- MIPS
- Microchip Technology PIC, (8-bit PIC16, PIC18, 16-bit dsPIC33 / PIC24), (32-bit PIC32)
- NXP Semiconductors LPC1000, LPC2000, LPC3000, LPC4000 (32-bit), LPC900, LPC700 (8-bit)
- Parallax Propeller
- PowerPC ISE
- Rabbit 2000 (8-bit)
- Renesas Electronics: RL78 16-bit MCU; RX 32-bit MCU; SuperH; V850 32-bit MCU; H8; R8C 16-bit MCU
- Silicon Laboratories Pipelined 8-bit 8051 microcontrollers and mixed-signal ARM-based 32-bit microcontrollers
- STMicroelectronics STM8 (8-bit), ST10 (16-bit), STM32 (32-bit), SPC5 (automotive 32-bit)
- Texas Instruments TI MSP430 (16-bit), MSP432 (32-bit), C2000 (32-bit)
- Toshiba TLCS-870 (8-bit/16-bit)

0.9 The History of Microcontroller

- On 21 June **2018**, the "world's smallest computer" was announced by the University of Michigan. The device is a "0.04mm3 16nW wireless and battery less sensor system with integrated Cortex-M0+ processor and optical communication for cellular temperature measurement." It "measures just 0.3 mm to a side—dwarfed by a grain of rice.





Introduction to Arduino

Introduction to Arduino

Contents

1.1 Objectives

- In this tutorial, you will learn:
  - To understand Arduino basic concepts.
  - To know about the Importance of an Arduino.
  - To become familiar with the types of Arduino Boards.
  - To understand the advantages and disadvantages of an Arduino.
  - To know how to start Arduino programming
  - To appreciate the history of an Arduino.

1.2 Introduction

- Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis had an idea for an easy-to-use microcontroller development board for Interaction Design Institute Ivrea in Ivrea, Italy way back **2005**. They need a cheap programmable device for students and artists. They selected the ATmega8 microcontroller, an AVR family of 8-bit microcontroller from Atmel, wrote bootloader firmware called Optiboot, and incorporated with Integrated Development Environment to write programs called "sketches." The project is called Arduino.

1.3 The History of an Arduino

- The Arduino project was started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy.At that time, the students used a BASIC Stamp microcontroller at a cost of $50, a considerable expense for many students. Standardization
- In 2003 Hernando Barragán created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas. Casey Reas is known for co-creating, with Ben Fry, the Processing development platform. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller.
- In 2005, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, extended Wiring by adding support for the cheaper ATmega8 microcontroller. The new project, forked from Wiring, was called Arduino.
- It was estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced,
- and in 2013 that 700,000 official boards were in users' hands.

1.4 What is an Arduino?

Arduino is an open-source embedded system development platform using C/C++ programming language with Arduino Libraries created by the Arduino Company and very popular in embedded community/users.

2 Essential parts of Arduino:

1. Arduino Development Board-the hardware part of the ecosystem
2. Arduino Integrated Development Environment (IDE)-the software part where you wrote codes called sketches to compile and upload to the hardware.

1.5 Why Is It Arduino Important?

- Easy-to-use programming environment
- Flexible enough for advanced users.
- Cross platform IDE, runs on Mac, Windows, and Linux.
- Low cost development board
- Free compiler
- Open source both hardware and software.
- Large Community Users

1.6 Types of Arduino Boards

1.7 Advantages and Disadvantages of an Arduino

Advantages of Arduino Platform compared to others:

1. Multiplatform environment and it can run on Windows, Mac and Linux.
2. IDE is based on Processing which is easy-to- use for both artists and engineers.
3. Uploading program via USB.PCs and laptops today has no serial ports.
4. It is open source hardware and software- you can copy and modify hardware and software for your own application as long as you published as General Public Licensed in the internet.
5. Huge community of users-there are so many Arduino libraries can be found at internet and you can add easily on IDE.
6. The learning curve is not that difficult-you can start immediately after reading the Arduino official website.

1.7 Advantages and Disadvantages of an Arduino

Disadvantages of Arduino Platform:

1. We don't get to get to the know about the microcontroller inside the **Arduino** deeply.

2. Also if we use **Arduino** IDE we get limited library. If we use Atmel studio we can know the microcontroller deeply.

3. Many people just copy the code from internet without any knowledge about it.

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

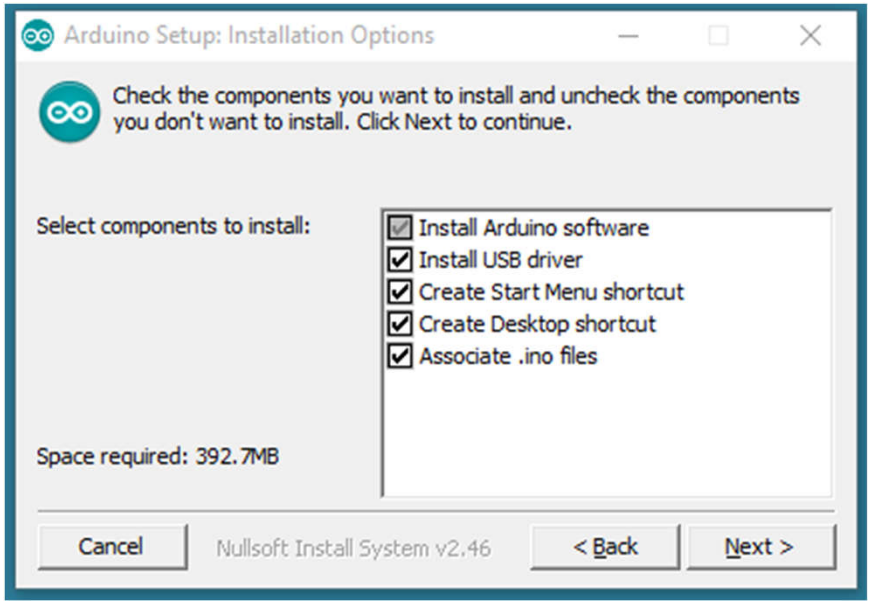
These are the steps how to connect Arduino Uno to your computer and upload your code called sketch. To upload the sketch to your Arduino Uno, you need to install the Arduino Software called Arduino Ingrated Development Environment (IDE). Download the installer using site below.

https://www.arduino.cc/download_handler.php?f=/arduino-1.8.12-windows.exe

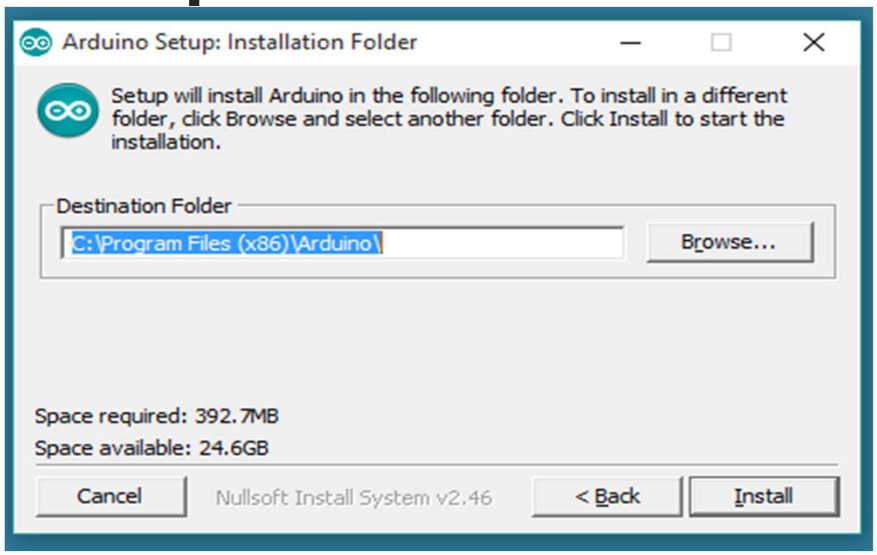
Or if you want standalone Arduino IDE you can also download in this link

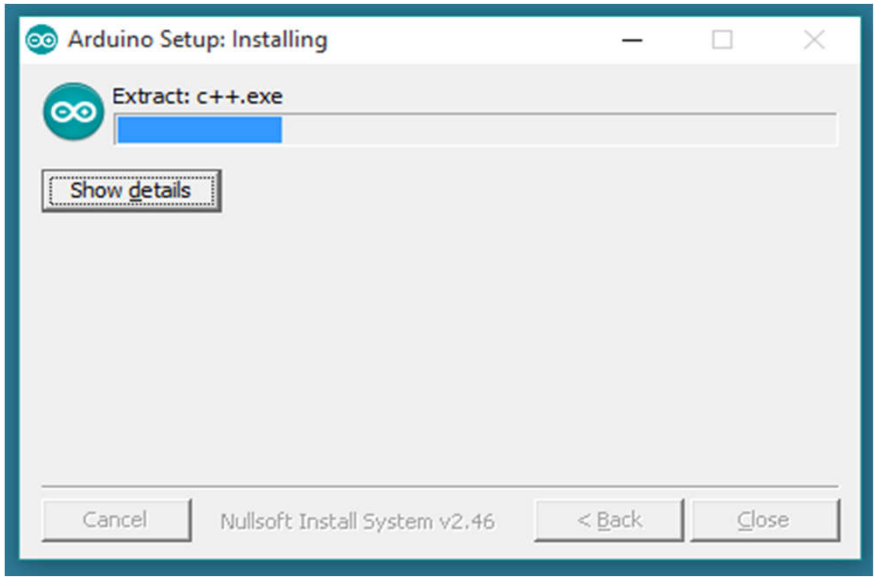
https://www.arduino.cc/download_handler.php?f=/arduino-1.8.12-windows.zip

During installaltion click all the components then select installation directory, wait until the Arduino IDE finished installed.

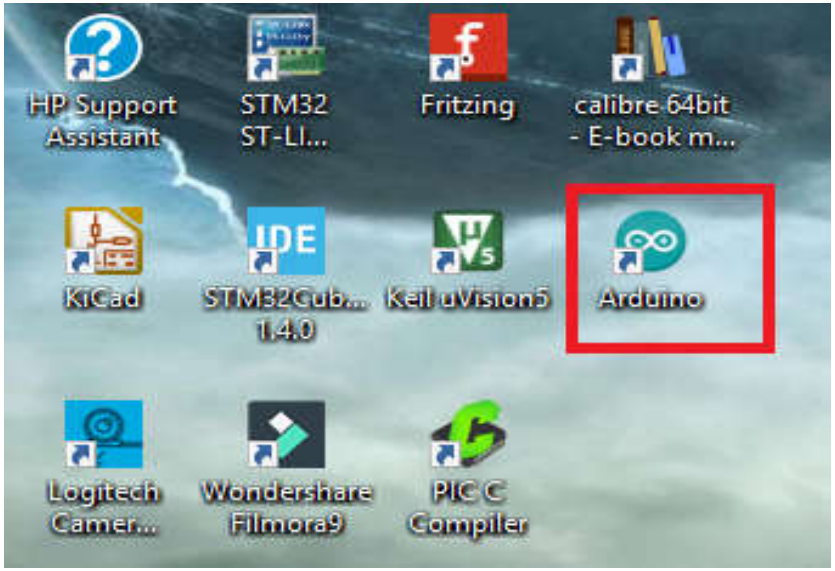


1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board





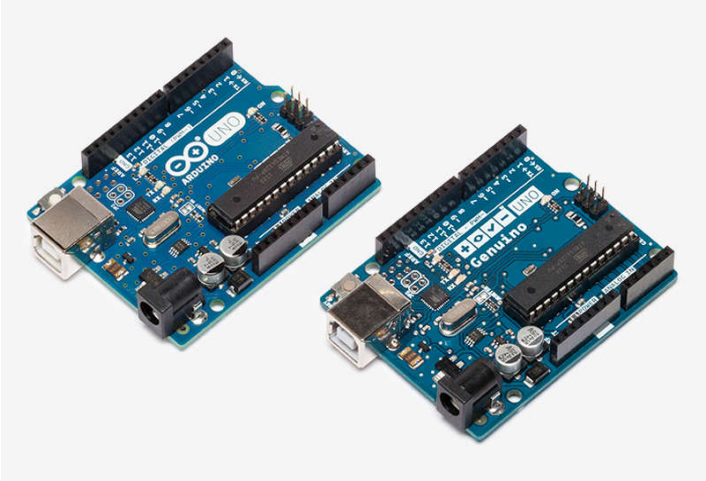
1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board



1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Installing the Arduino Drivers

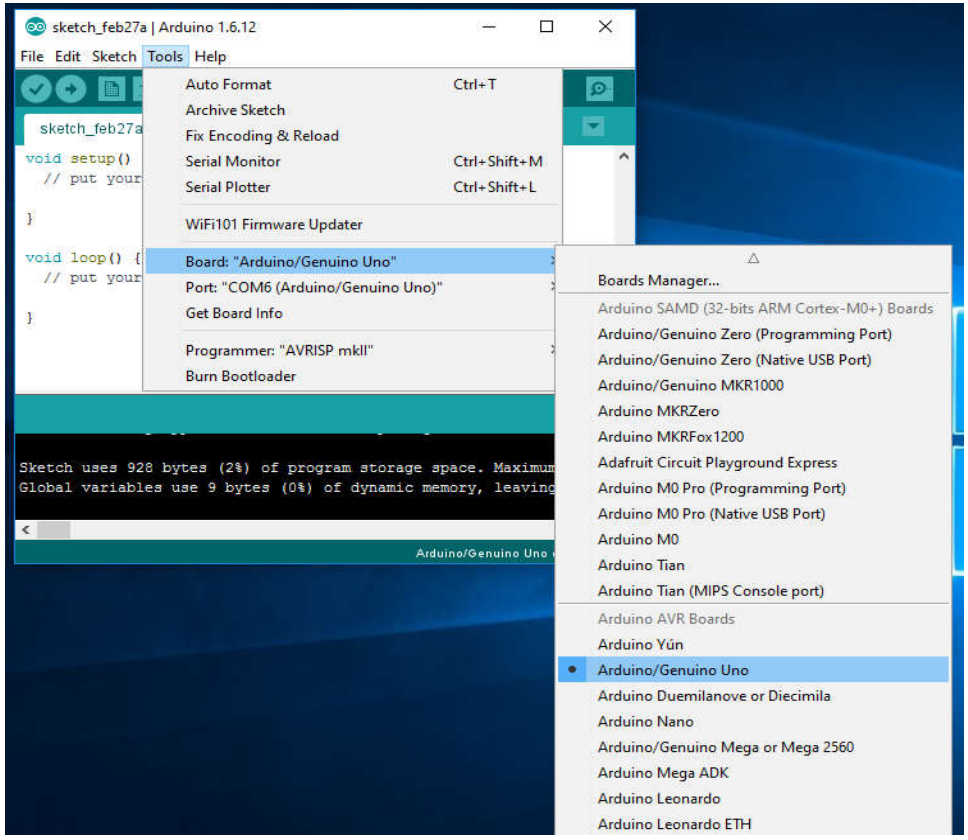
By using USB cable, connect the Arduino Uno board to the PC or laptop



1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

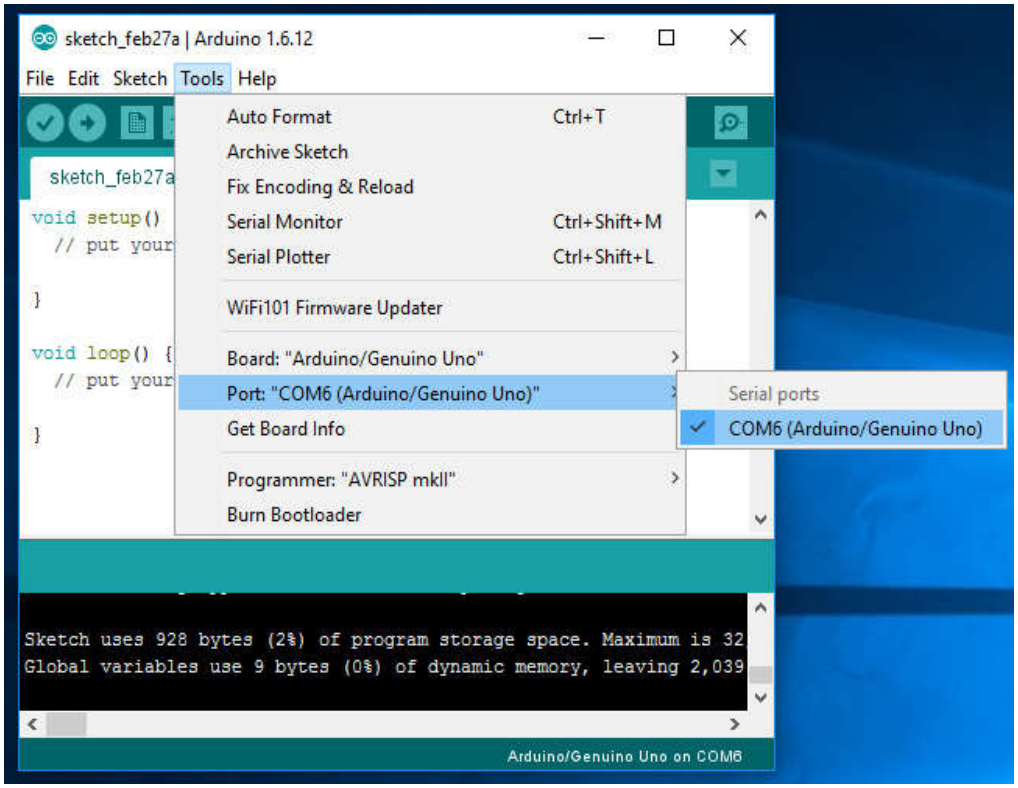
Testing your first program the Hello World of Embedded System: Blinking LED

1. Open the Arduino IDE and connect the Arduino Uno to the computer.
2. Select Tools->Board then choose Arduino/Genuino Uno.



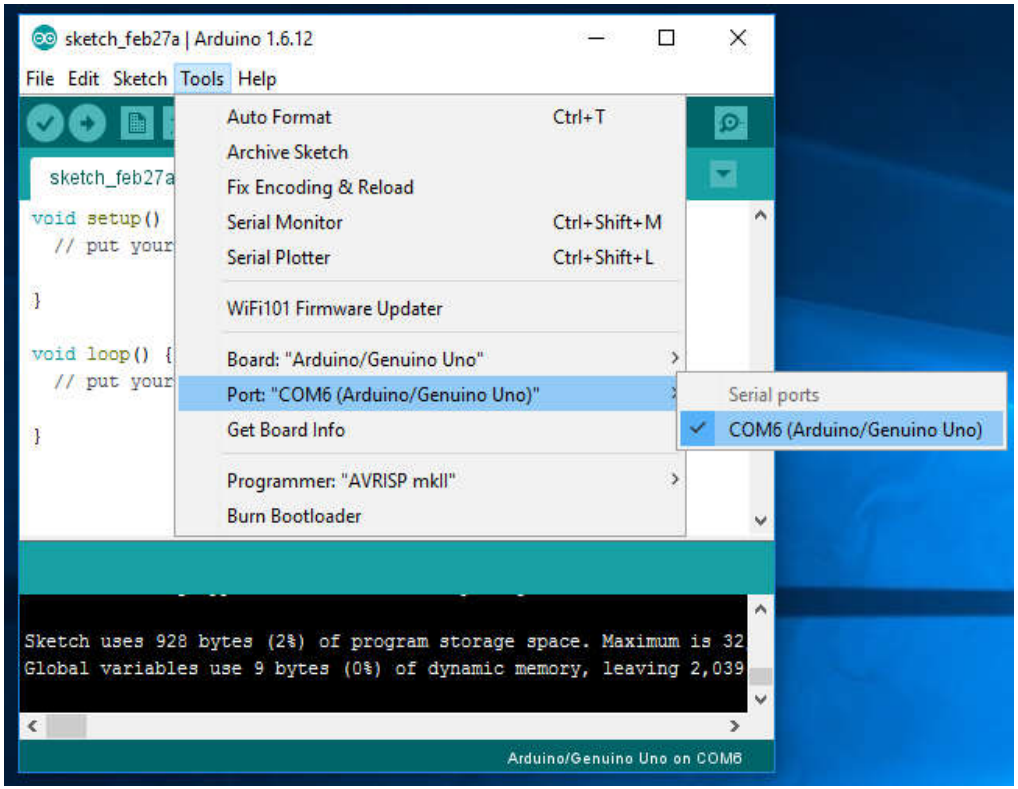
1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

3. Click Tools->Port the select COM port of Arduino/Genuino Uno.



1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

4. Next is go to File->Examples->Basics then Blink. The IDE will display another window and the sketch Blink is already encoded in the IDE. See the code below.

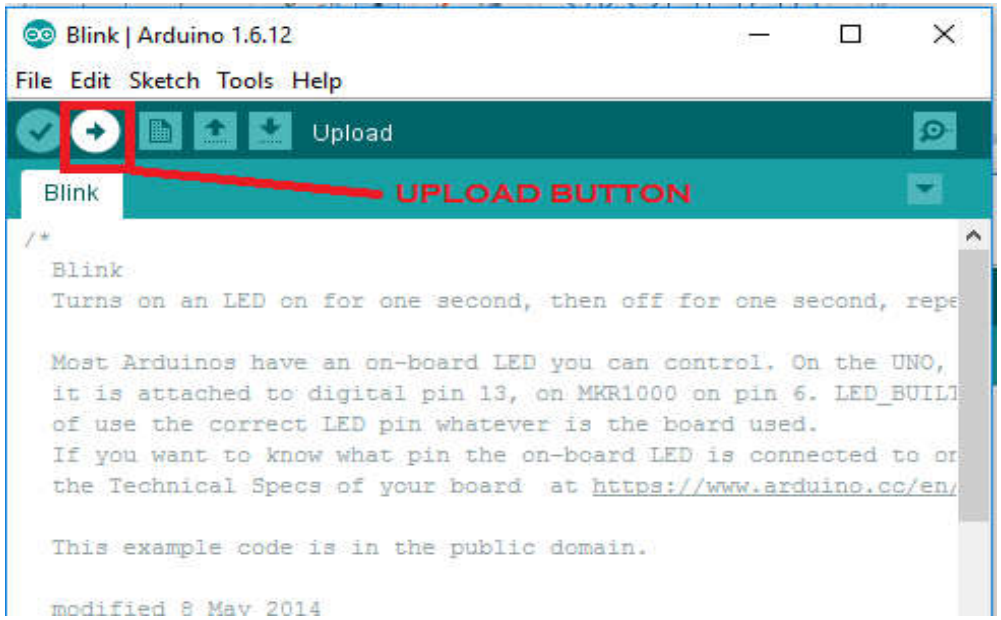


1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN takes care
of use the correct LED pin whatever is the board used.
If you want to know what pin the on-board LED is connected to on your Arduino model,
check
*/
// the setup function runs once when you press reset or power the board
void setup()
{
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}

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

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

5. Click the Upload button in the IDE and wait after the code or sketch uploaded into the Arduino Uno.



6. After programming the Arduino Uno, the pin 13 with LED is blinking ON and OFF for a second.

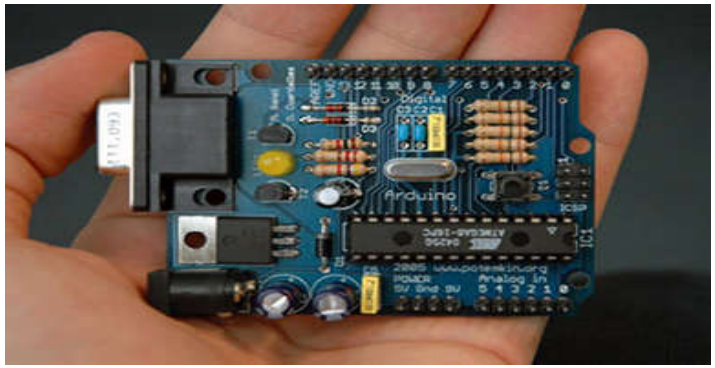
1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

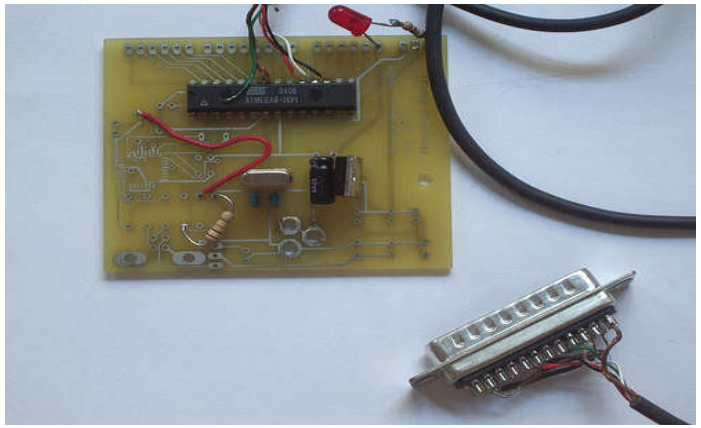
Technical specs of an Arduino Uno

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**
LED_BUILTIN	**13**

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

- A project which began in Ivrea (Italy) in 2005
- Massimo Banzi
- David Cuartielles
- device for controlling student-built interaction design
- less expensive
- more modern than what is available
- Basic Stamp killer





1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

What is a Arduino?

- named the project after a local bar
- italian masculine first name
- "strong friend"
- English equivalent is "Hardwin"

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Setup and Loop

```
void setup()
{
   //enter code here
}

void loop()
{
   //enter code here
 }
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Setup()

called when a sketch starts running

use it to initialize

variables,

pin modes,

- start using libraries
- will only run once, after each powerup or reset of the Arduino board

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Setup()

```
void setup()
    {
        LEDPin = 13;
        pinMode(LEDPin, OUTPUT);
        digitalWrite(LEDPin, LOW);
    }

    void loop()
    {
    }
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Loop()

-Executed repeatedly

-Allows the program to change and respond.

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Loop()

```
void setup()
{
    LEDPin = 13;
    pinMode(LEDPin, OUTPUT);
    digitalWrite(LEDPin, LOW);
}

void loop()
{
    delay(500);
    digitalWrite(LEDPin, HIGH);
    delay(500);
    digitalWrite(LEDPin, LOW);
}
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- Predefined functions to easily configure Arduino
- Pin Operation
- Peripheral Operation

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- pinMode(pin, mode)
- Configures a digital pin to behave either as an input or an output
- pin: the pin number whose mode you wish to set
- mode: either INPUT or OUTPUT

- Example

```
pinMode(13, OUTPUT);
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- digitalWrite(pin, value)
- Write a HIGH or a LOW value to a digital pin
- pin: the pin number whose value you wish to set
- value: either HIGH or LOW

- Example
  digitalWrite(13, HIGH);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- digitalRead(pin)
- Reads the value from a specified digital pin
- pin: the digital pin number you want to read
- returns either HIGH or LOW

- Example
- result = digitalRead(13);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- delay(value)
- Pauses the program for the amount of time
- value: the number of milliseconds to pause

- Example
- delay(1000);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- analogRead(pin)
- Reads the value from a specified analog pin
- pin: the analog pin number you want to read
- returns an integer (0 to 1023) corresponding to pin voltage

- Example
- result = analogRead(13);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

```
Blinking LED Example

int LEDPin;
// setup initializes the LED pin as output
// and initially turned off
void setup()
{
        LEDPin = 13;
        pinMode(LEDPin, OUTPUT);
        digitalWrite(LEDPin, LOW);
}
// loop checks the button pin each time,
void loop()
{
        delay(500);
        digitalWrite(LEDPin, HIGH);
        delay(500);
        digitalWrite(LEDPin, LOW);
}
```

1.9 Arduino Language Reference

Digital I/O

digitalWrite()

Description
Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(),
its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.
It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim.
Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor,
which acts like a large current-limiting resistor.

Syntax:
digitalWrite(pin, value)

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

Returns:
Nothing

1.9 Arduino Language Reference

Example Code:
The code makes the digital pin 13 an OUTPUT and toggles it by alternating between HIGH and LOW at one second pace.

```
void setup()
{
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

1.9 Arduino Language Reference

digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW.

Syntax
digitalRead(pin)

Returns
HIGH or LOW

Example Code
Sets pin 13 to the same value as pin 7, declared as an input.

```
int ledPin = 13;  // LED connected to digital pin 13
int inPin = 7;    // pushbutton connected to digital pin 7
int val = 0;      // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);  // sets the digital pin 13 as output
  pinMode(inPin, INPUT);    // sets the digital pin 7 as input
}

void loop()
{
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);  // sets the LED to the button's value
}
```

1.9 Arduino Language Reference

pinMode()

Description
Configures the specified pin to behave either as an input or an output.

Syntax
pinMode(pin, mode)

Parameters
pin: the Arduino pin number to set the mode of.
mode: INPUT, OUTPUT, or INPUT_PULLUP.

Returns
Nothing

Example Code
The code makes the digital pin 13 OUTPUT and Toggles it HIGH and LOW

```
void setup()
{
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

1.9 Arduino Language Reference

Analog I/O

Description
Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023.
On an Arduino UNO, for example,
this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

The input range can be changed using analogReference(), while the resolution can be changed (only for Zero, Due and MKR boards) using analogReadResolution().

On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

1.9 Arduino Language Reference

Syntax
analogRead(pin)

Parameters
pin: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns
The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits).
Data type: int.

Example Code
The code reads the voltage on analogPin and displays it.

```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0;  // variable to store the value read

void setup()
{
  Serial.begin(9600);           //  setup serial
}

void loop()
{
  val = analogRead(analogPin);  // read the input pin
  Serial.println(val);          // debug value
}
```

1.9 Arduino Language Reference

analogReference()

Description
Configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The options are:

Arduino AVR Boards (Uno, Mega, Leonardo, etc.)

DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)

INTERNAL: a built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328P and 2.56 volts on the ATmega32U4 and ATmega8 (not available on the Arduino Mega)

INTERNAL1V1: a built-in 1.1V reference (Arduino Mega only)

INTERNAL2V56: a built-in 2.56V reference (Arduino Mega only)

EXTERNAL: the voltage applied to the AREF pin (0 to 5V only) is used as the reference.

1.9 Arduino Language Reference

Syntax
analogReference(type)

Parameters
type: which type of reference to use (see list of options in the description).

Returns
Nothing

Notes and Warnings
After changing the analog reference, the first few readings from analogRead() may not be accurate.

Don't use anything less than 0V or more than 5V for external reference voltage on the AREF pin! If you're using an external reference on the AREF pin, you must set the analog reference to EXTERNAL before calling analogRead(). Otherwise, you will short together the active reference voltage (internally generated) and the AREF pin, possibly damaging the microcontroller on your Arduino board.

Alternatively, you can connect the external reference voltage to the AREF pin through a 5K resistor, allowing you to switch between external and internal reference voltages. Note that the resistor will alter the voltage that gets used as the reference because there is an internal 32K resistor on the AREF pin. The two act as a voltage divider, so, for example, 2.5V applied through the resistor will yield 2.5 * 32 / (32 + 5) = ~2.2V at the AREF pin.

1.9 Arduino Language Reference

Random Numbers

random()

Description
The random function generates pseudo-random numbers.

Syntax
random(max)
random(min, max)

Parameters
min: lower bound of the random value, inclusive (optional).
max: upper bound of the random value, exclusive.

Returns
A random number between min and max-1.
Data type: long.

1.9 Arduino Language Reference

Example Code
The code generates random numbers and displays them.

```
long randNumber;

void setup() {
  Serial.begin(9600);

  // if analog input pin 0 is unconnected, random analog
  // noise will cause the call to randomSeed() to generate
  // different seed numbers each time the sketch runs.
  // randomSeed() will then shuffle the random function.
  randomSeed(analogRead(0));
}

void loop() {
  // print a random number from 0 to 299
  randNumber = random(300);
  Serial.println(randNumber);

  // print a random number from 10 to 19
  randNumber = random(10, 20);
  Serial.println(randNumber);

  delay(50);
}
```

1.9 Arduino Language Reference

randomSeed()

Description
randomSeed() initializes the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence. This sequence, while very long, and random, is always the same.

If it is important for a sequence of values generated by random() to differ, on subsequent executions of a sketch, use randomSeed() to initialize the random number generator with a fairly random input, such as analogRead() on an unconnected pin.

Conversely, it can occasionally be useful to use pseudo-random sequences that repeat exactly. This can be accomplished by calling randomSeed() with a fixed number, before starting the random sequence.

Syntax
randomSeed(seed)

Parameters
seed: number to initialize the pseudo-random sequence. Allowed data types: unsigned long.

Returns
Nothing

1.9 Arduino Language Reference

Example Code
The code generates a pseudo-random number and sends the generated number to the serial port.

```
long randNumber;

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop() {
  randNumber = random(300);
  Serial.println(randNumber);
  delay(50);
}
```

1.9 Arduino Language Reference

Timers

delay()

Description
Pauses the program for the amount of time (in milliseconds) specified as parameter.
(There are 1000 milliseconds in a second.)

Syntax
delay(ms)

Parameters
ms: the number of milliseconds to pause. Allowed data types: unsigned long.

Returns
Nothing

Example Code
The code pauses the program for one second before toggling the output pin.

```
int ledPin = 13;                // LED connected to digital pin 13

void setup() {
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                  // waits for a second
}
```

1.9 Arduino Language Reference

delayMicroseconds()

Description
Pauses the program for the amount of time (in microseconds) specified by the parameter. There are a thousand microseconds in a millisecond and a million microseconds in a second.

Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay() instead.

Syntax
delayMicroseconds(us)

Parameters
us: the number of microseconds to pause. Allowed data types: unsigned int.

Returns
Nothing

Example Code
The code configures pin number 8 to work as an output pin. It sends a train of pulses of approximately 100 microseconds period. The approximation is due to execution of the other instructions in the code.

```
int outPin = 8;                 // digital pin 8

void setup() {
  pinMode(outPin, OUTPUT);      // sets the digital pin as output
}

void loop() {
  digitalWrite(outPin, HIGH); // sets the pin on
  delayMicroseconds(50);      // pauses for 50 microseconds
  digitalWrite(outPin, LOW);  // sets the pin off
  delayMicroseconds(50);      // pauses for 50 microseconds
}
```


1.9 Arduino Language Reference

millis()

Description
Returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Syntax
time = millis()

Parameters
None

Returns
Number of milliseconds passed since the program started. Data type: unsigned long.

Example Code
This example code prints on the serial port the number of milliseconds passed since the Arduino board started running the code itself.

```
unsigned long myTime;

void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.print("Time: ");
  myTime = millis();

  Serial.println(myTime); //prints time since program started
  delay(1000);          // wait a second so as not to send massive amounts of data
}
```

1.9 Arduino Language Reference

Advanced I/O

noTone()

Description
Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

Syntax
noTone(pin)

Parameters
pin: the Arduino pin on which to stop generating the tone

Returns
Nothing

Note:
If you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

1.9 Arduino Language Reference

tone()

Description
Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

It is not possible to generate tones lower than 31Hz. For technical details, see Brett Hagman's notes.

Syntax
tone(pin, frequency)
tone(pin, frequency, duration)

Parameters
pin: the Arduino pin on which to generate the tone.
frequency: the frequency of the tone in hertz. Allowed data types: unsigned int.
duration: the duration of the tone in milliseconds (optional). Allowed data types: unsigned long.

Returns
Nothing

Notes
If you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

1.9 Arduino Language Reference

pulseIn()

Description
Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

The timing of this function has been determined empirically and will probably show errors in longer pulses. Works on pulses from 10 microseconds to 3 minutes in length.

Syntax
pulseIn(pin, value)
pulseIn(pin, value, timeout)

Parameters
pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. Allowed data types: unsigned long.

Returns
The length of the pulse (in microseconds) or 0 if no pulse started before the timeout. Data type: unsigned long.

1.9 Arduino Language Reference

Example Code
The example prints the time duration of a pulse on pin 7.

```
int pin = 7;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseIn(pin, HIGH);
  Serial.println(duration);
}
```

1.9 Arduino Language Reference

pulseInLong()

Description
pulseInLong() is an alternative to pulseIn() which is better at handling long pulse and interrupt affected scenarios.

Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseInLong() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

The timing of this function has been determined empirically and will probably show errors in shorter pulses. Works on pulses from 10 microseconds to 3 minutes in length. This routine can be used only if interrupts are activated. Furthermore the highest resolution is obtained with large intervals.

Syntax
pulseInLong(pin, value)
pulseInLong(pin, value, timeout)

Parameters
pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. Allowed data types: unsigned long.

Returns
The length of the pulse (in microseconds) or 0 if no pulse started before the timeout. Data type: unsigned long.

1.9 Arduino Language Reference

Example Code
The example prints the time duration of a pulse on pin 7.

```
int pin = 7;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseInLong(pin, HIGH);
  Serial.println(duration);
}
```

Notes
This function relies on micros() so cannot be used in noInterrupts() context.

1.9 Arduino Language Reference

shiftIn()

Description
Shifts in a byte of data one bit at a time. Starts from either the most (i.e. the leftmost) or least (rightmost) significant bit. For each bit, the clock pin is pulled high, the next bit is read from the data line, and then the clock pin is taken low.

If you're interfacing with a device that's clocked by rising edges, you'll need to make sure that the clock pin is low before the first call to shiftIn(), e.g. with a call to digitalWrite(clockPin, LOW).

Note: this is a software implementation; Arduino also provides an SPI library that uses the hardware implementation, which is faster but only works on specific pins.

Syntax
byte incoming = shiftIn(dataPin, clockPin, bitOrder)

Parameters
dataPin: the pin on which to input each bit. Allowed data types: int.
clockPin: the pin to toggle to signal a read from dataPin.
bitOrder: which order to shift in the bits; either MSBFIRST or LSBFIRST. (Most Significant Bit First, or, Least Significant Bit First).

Returns
The value read. Data type: byte.

1.9 Arduino Language Reference

shiftOut()

Description
Shifts out a byte of data one bit at a time. Starts from either the most (i.e. the leftmost) or least (rightmost) significant bit. Each bit is written in turn to a data pin, after which a clock pin is pulsed (taken high, then low) to indicate that the bit is available.

Note- if you're interfacing with a device that's clocked by rising edges, you'll need to make sure that the clock pin is low before the call to shiftOut(), e.g. with a call to digitalWrite(clockPin, LOW).

This is a software implementation; see also the SPI library, which provides a hardware implementation that is faster but works only on specific pins.

Syntax
shiftOut(dataPin, clockPin, bitOrder, value)

Parameters
dataPin: the pin on which to output each bit. Allowed data types: int.
clockPin: the pin to toggle once the dataPin has been set to the correct value. Allowed data types: int.
bitOrder: which order to shift out the bits; either MSBFIRST or LSBFIRST. (Most Significant Bit First, or, Least Significant Bit First).
value: the data to shift out. Allowed data types: byte.

Returns
Nothing

1.9 Arduino Language Reference

Example Code
For accompanying circuit, see the tutorial on controlling a 74HC595 shift register.

```
//  Name    : shiftOutCode, Hello World                   //
//  Author  : Carlyn Maw,Tom Igoe                         //
//  Date    : 25 Oct, 2006                                //
//  Version : 1.0                                         //
//  Notes   : Code for using a 74HC595 Shift Register     //
//          : to count from 0 to 255                      //
//Pin connected to ST_CP of 74HC595
int latchPin = 8;
//Pin connected to SH_CP of 74HC595
int clockPin = 12;
////Pin connected to DS of 74HC595
int dataPin = 11;

void setup() {
  //set pins to output because they are addressed in the main loop
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  //count up routine
  for (int j = 0; j < 256; j++) {
    //ground latchPin and hold low for as long as you are transmitting
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, j);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(latchPin, HIGH);
    delay(1000);
  }
}
```

1.9 Arduino Language Reference

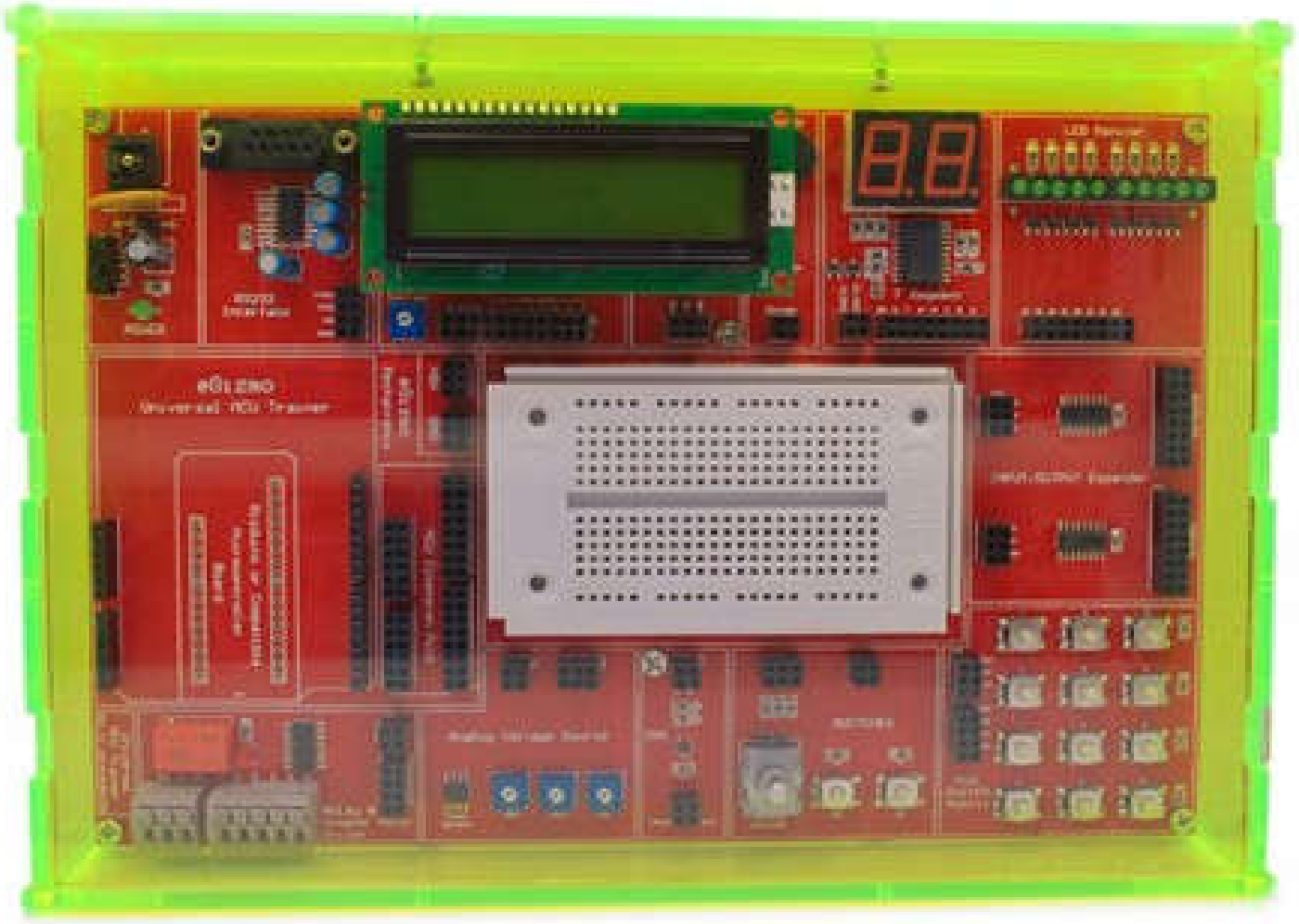
Notes
The dataPin and clockPin must already be configured as outputs by a call to pinMode().

shiftOut is currently written to output 1 byte (8 bits) so it requires a two step operation to output values larger than 255.

```
// Do this for MSBFIRST serial
int data = 500;
// shift out highbyte
shiftOut(dataPin, clock, MSBFIRST, (data >> 8));
// shift out lowbyte
shiftOut(dataPin, clock, MSBFIRST, data);

// Or do this for LSBFIRST serial
data = 500;
// shift out lowbyte
shiftOut(dataPin, clock, LSBFIRST, data);
// shift out highbyte
shiftOut(dataPin, clock, LSBFIRST, (data >> 8));
```

1.10 Arduino Preferred Kits

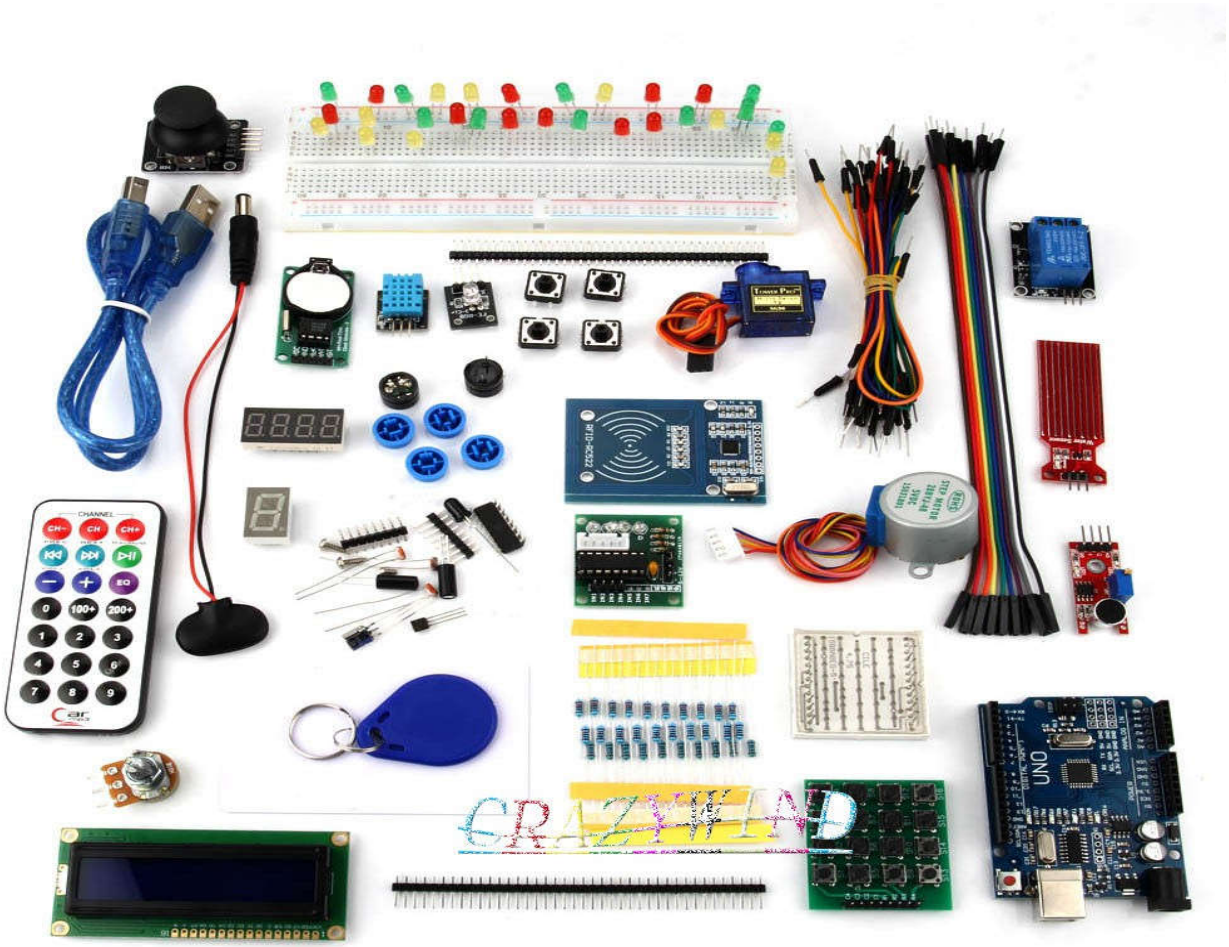
Universal MCU Digital Trainer



https://www.e-gizmo.net/oc/index.php?route=product/product&product_id=481

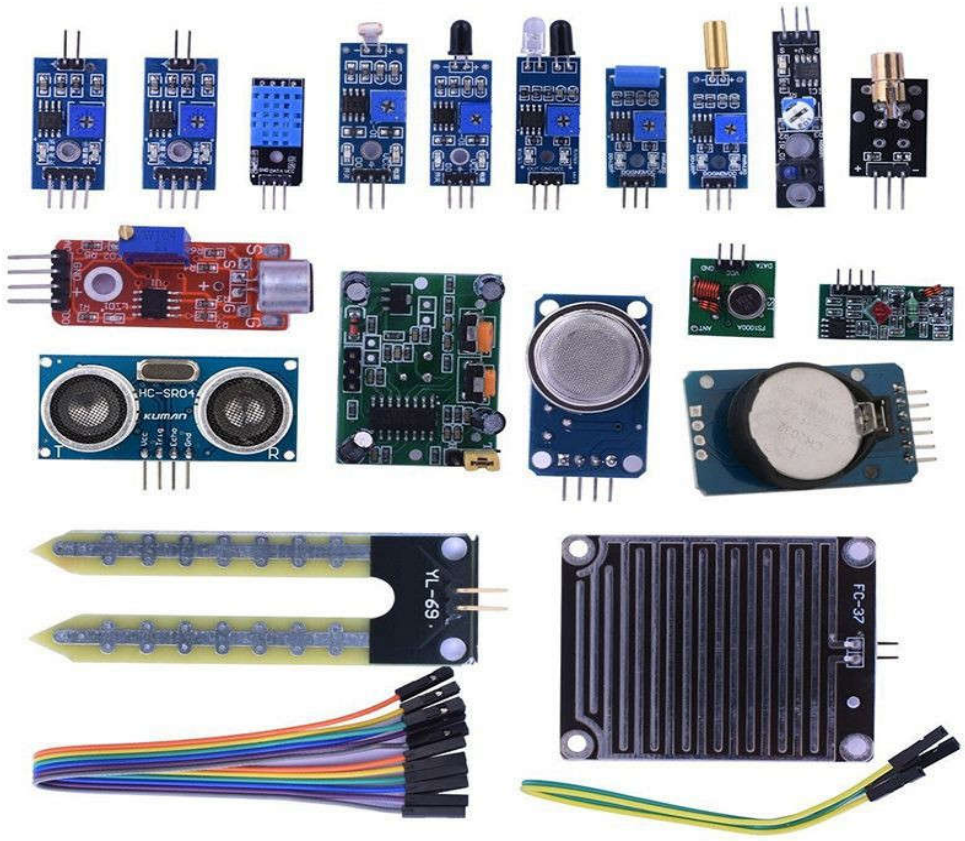
1.10 Preferred Kits for Arduino

RFID Learning Starter Kit for Arduino R3 Upgraded Version



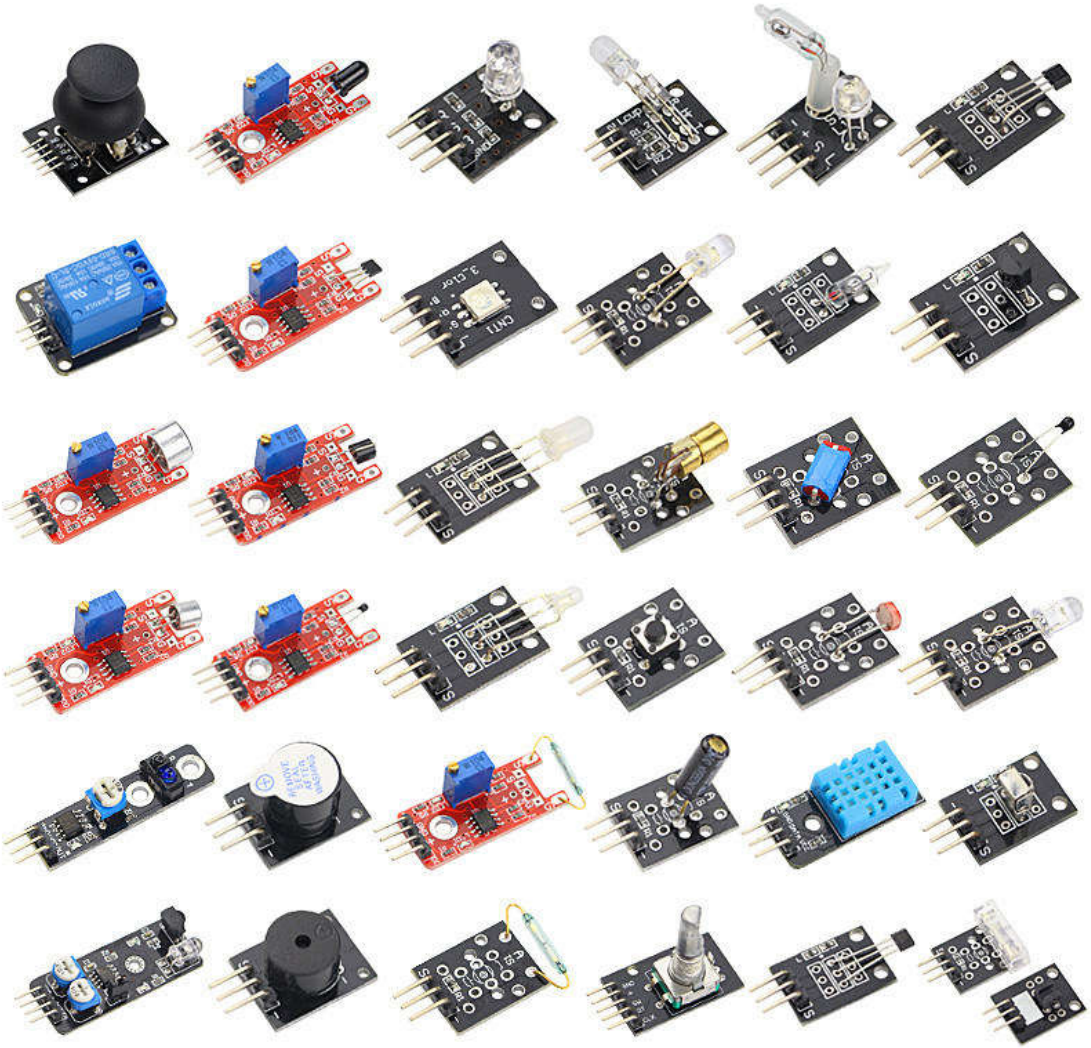
1.10 Preferred Kits for Arduino

16 in 1 Sensor Kit



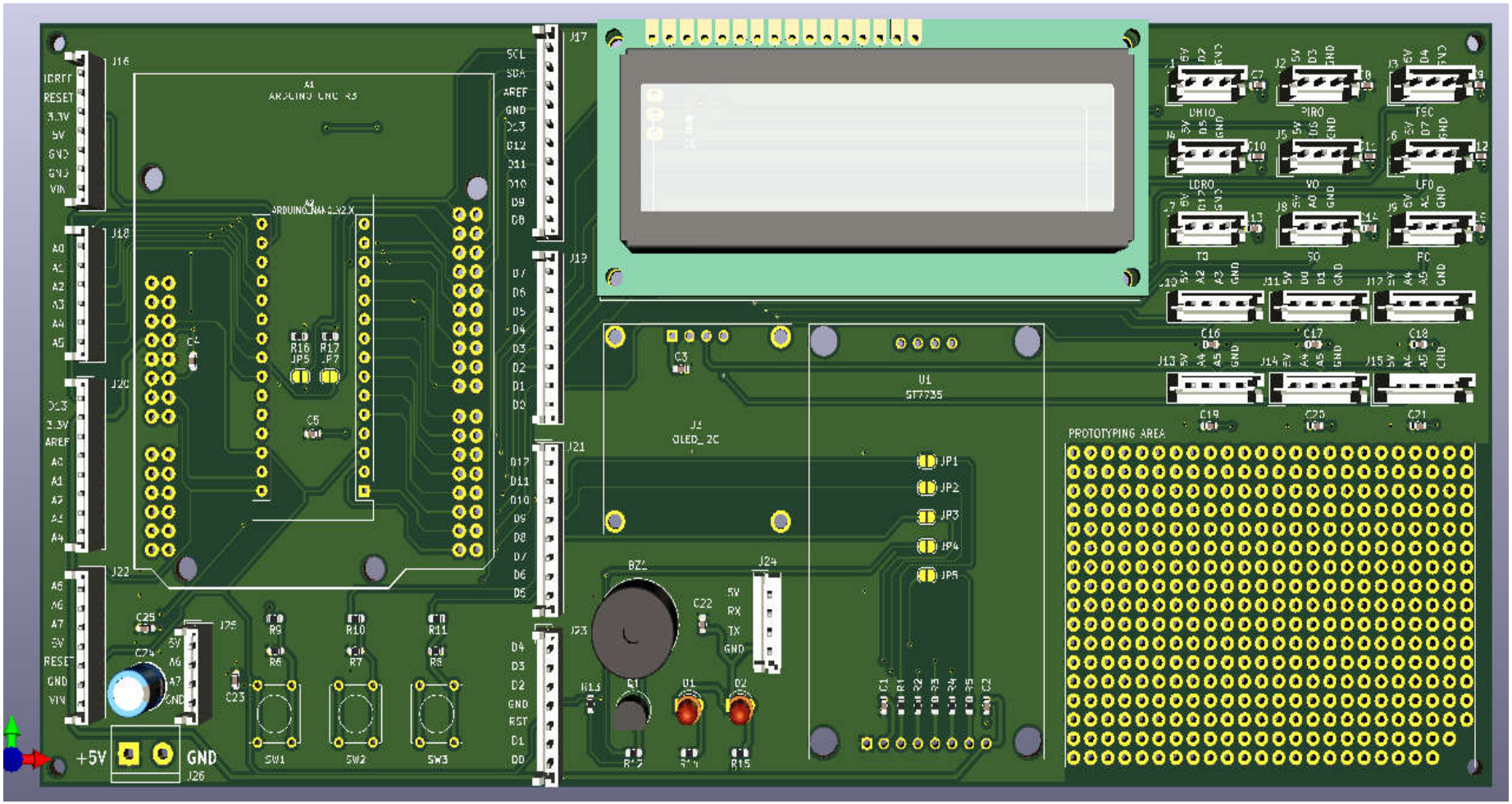
1.10 Preferred Kits for Arduino

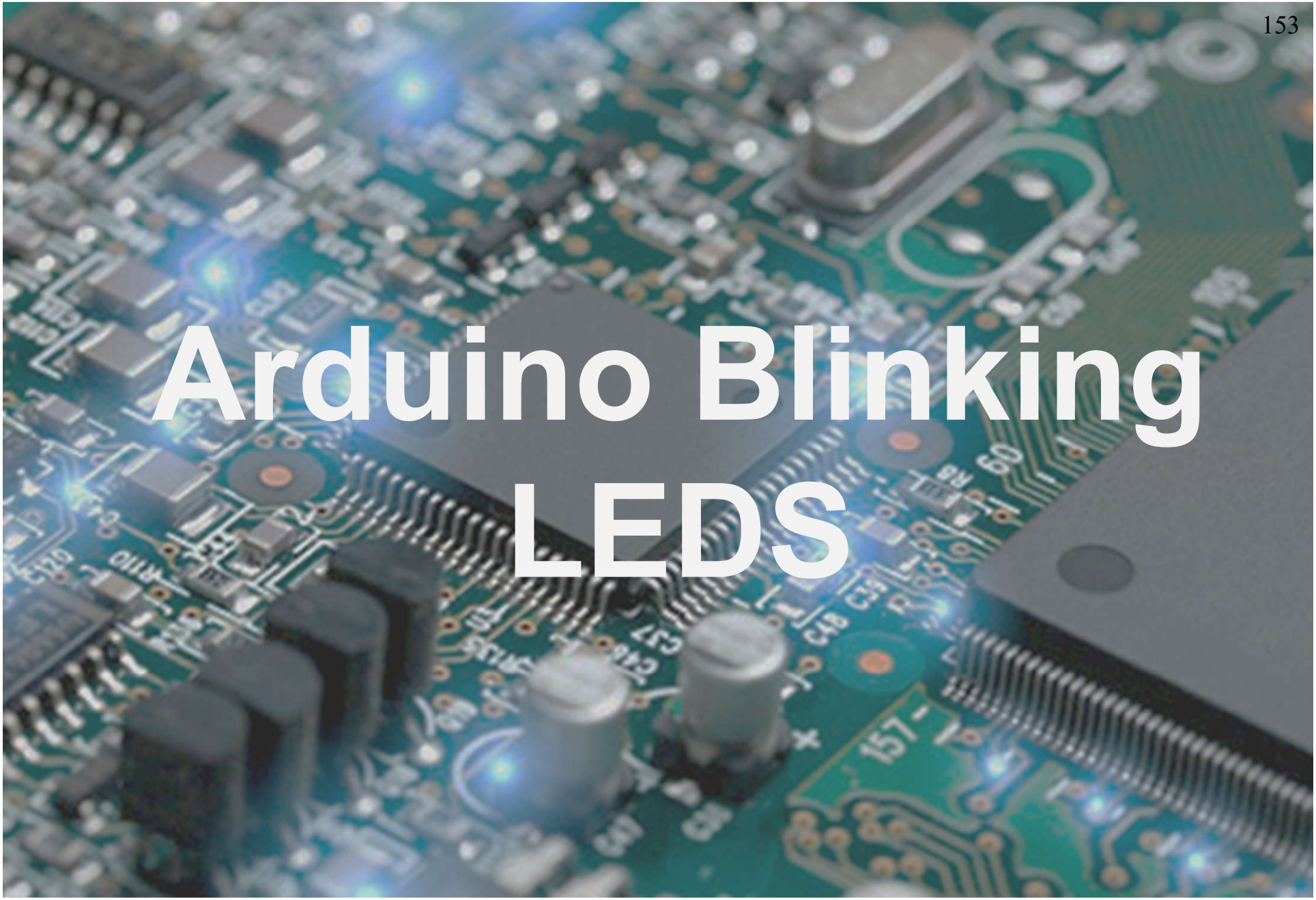
37 in 1 Sensor Kit



1.10 Preferred Kits for Arduino

Display Technology Kit Soon…





Arduino Blinking LEDS

Arduino Blinking LEDs

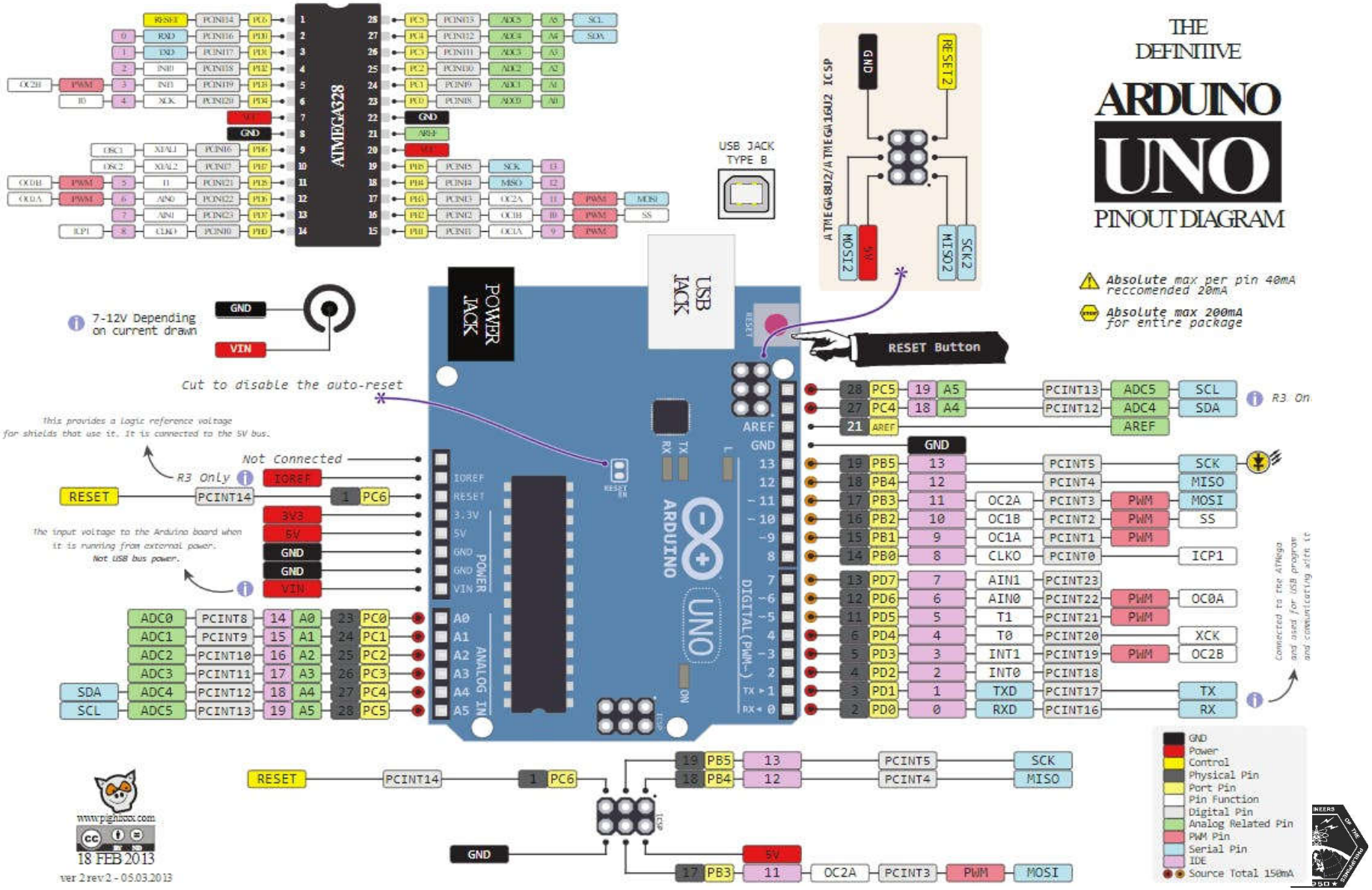
Contents

2.0 Objectives

- In this tutorial, you will learn:
  - To understand Arduino basic Inputs and Outputs (I/O's).
  - To know about the Importance of an Light Emitting Diodes (LEDs).
  - To become familiar with the types of Resistors.
  - To understand the difference between Pull-up and Pull-down in Buttons.
  - To know how to use Breadboard and Jumper Wires
  - To be able to start Arduino Programming using Blinking LED Project: "Hello World" of embedded systems.
  - To appreciate how the Blinking LED code works

2.1 Arduino I/O's

- Arduino Pinouts Guide



2.1 Arduino I/O's

- **Arduino I/O's**

Inputs:

Arduino pins are configured as INPUT by default.

Pins are configured in High-Impedance state typically >100Mohm impedance.

Outputs:

Pin configured as output is in low impedance state to drive as an output to other circuit. Atmega pins can be **source** (positive current source) or **sink** (negative current source) up to 40mA max.

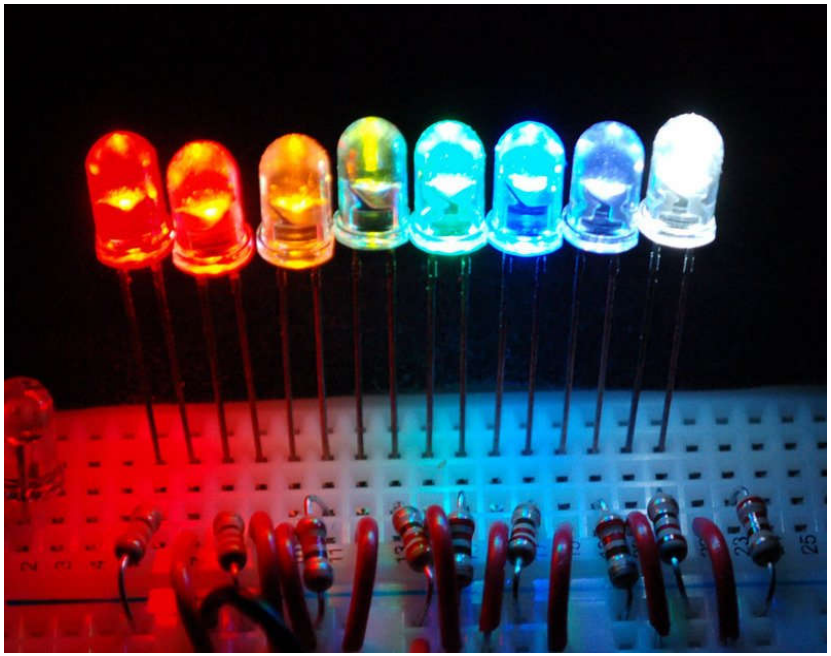
2.1 Arduino I/O's

Technical Specifications:

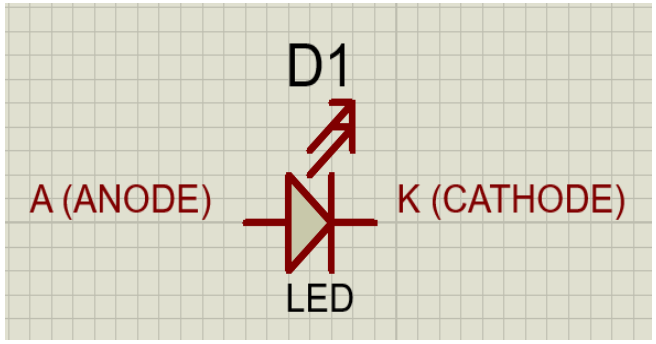
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

2.2 Light Emitting Diodes

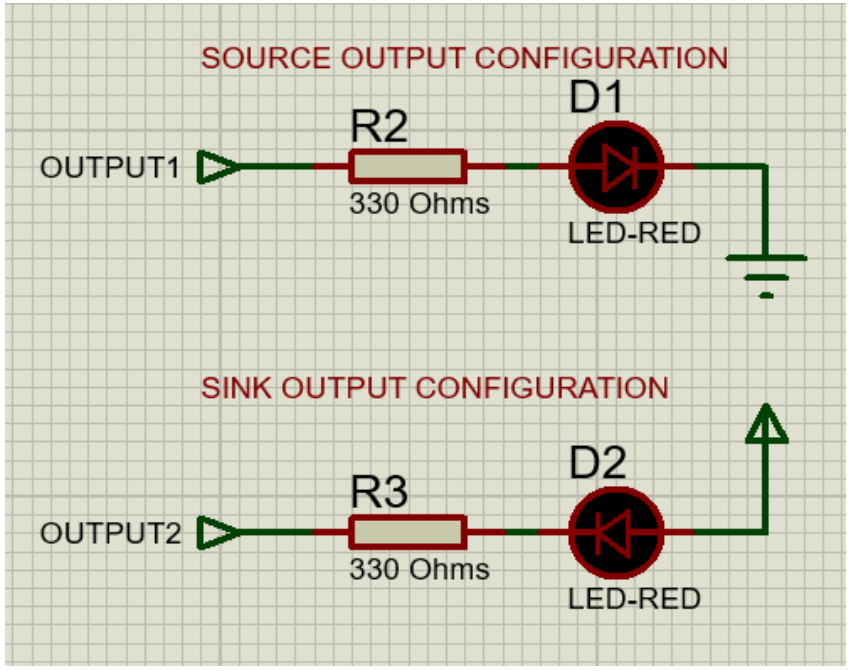
- **Light-emitting diode (LED)** is a semiconductor light source that emits light when current flows through it.
- In 1962, **Nick Holonyak Jr.** is an American engineer who invented the first LED that emitted visible red light while working at General Electric laboratory in New York.



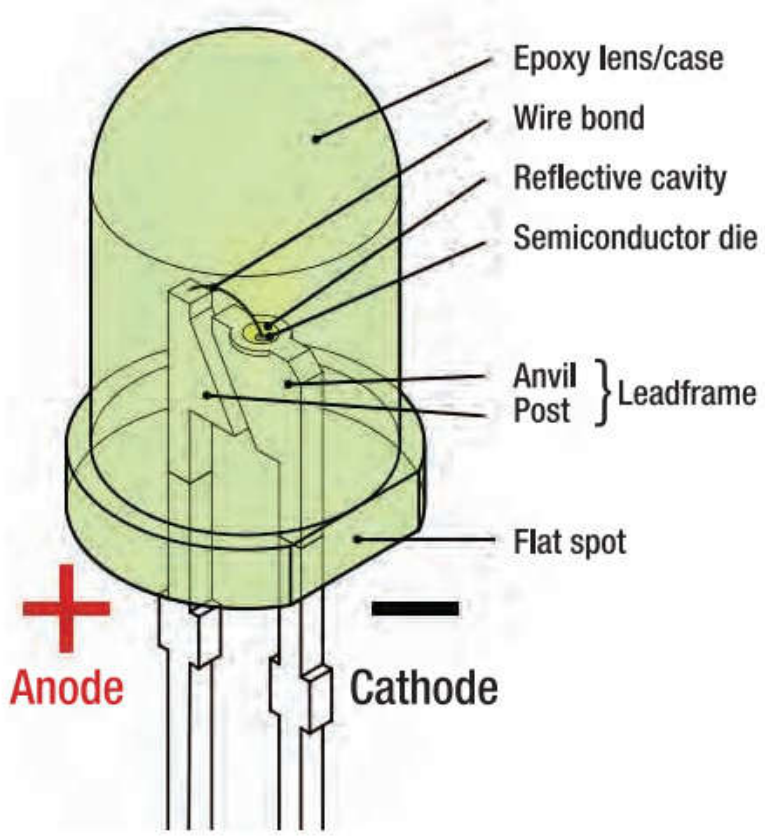
2.2 Light Emitting Diodes



LED SYMBOL



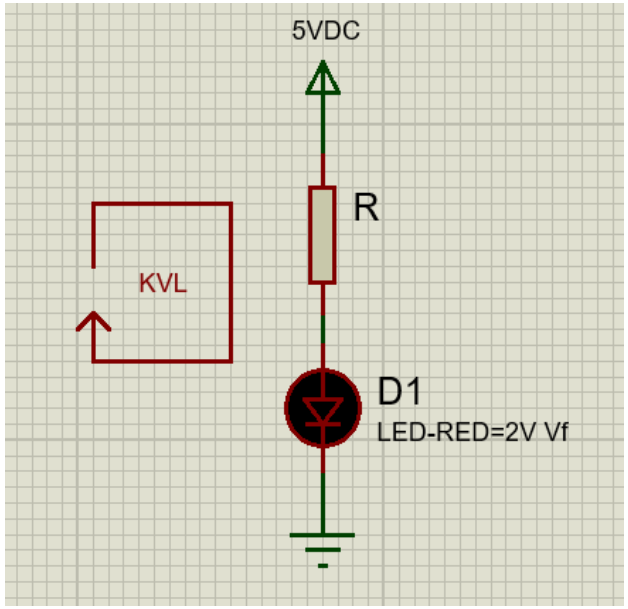
SOURCE/SINK OUTPUT TYPE



2.2 Light Emitting Diodes

Typical LED has a maximum forward current $\mathbf{I_f}$ has 25-30mA, but for simplicity, we use 10mA for nominal forward current and 2V for nominal forward voltage $\mathbf{V_f}$, so from KVL, 5-I*R-2=0; R = (5-2/10mA) thus:

R= 300 Ohms or 330 Ohms for common usage



Solving the Value of R

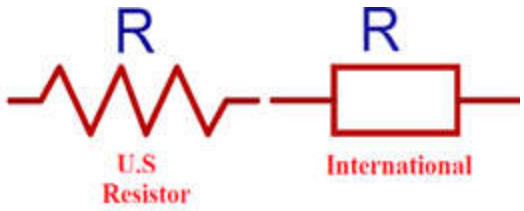
Colour	Approx. Forward Voltage Vf (V)
Red	1.7
HE Red*	2.0
Bright Red	2.3
Orange	2.0
Yellow	2.1
Green	2.2
Blue	3.2
White	3.2

*HE - High Efficiency

Typical LED Forward Voltage

2.3 Resistors

- **Resistor** is a passive two-terminal electrical component that implements electrical resistance to reduce the rate of flow of electrons (electrical current). The unit of resistance is Ohms and has a symbol of omega Ω.



- **Applications of Resistor**

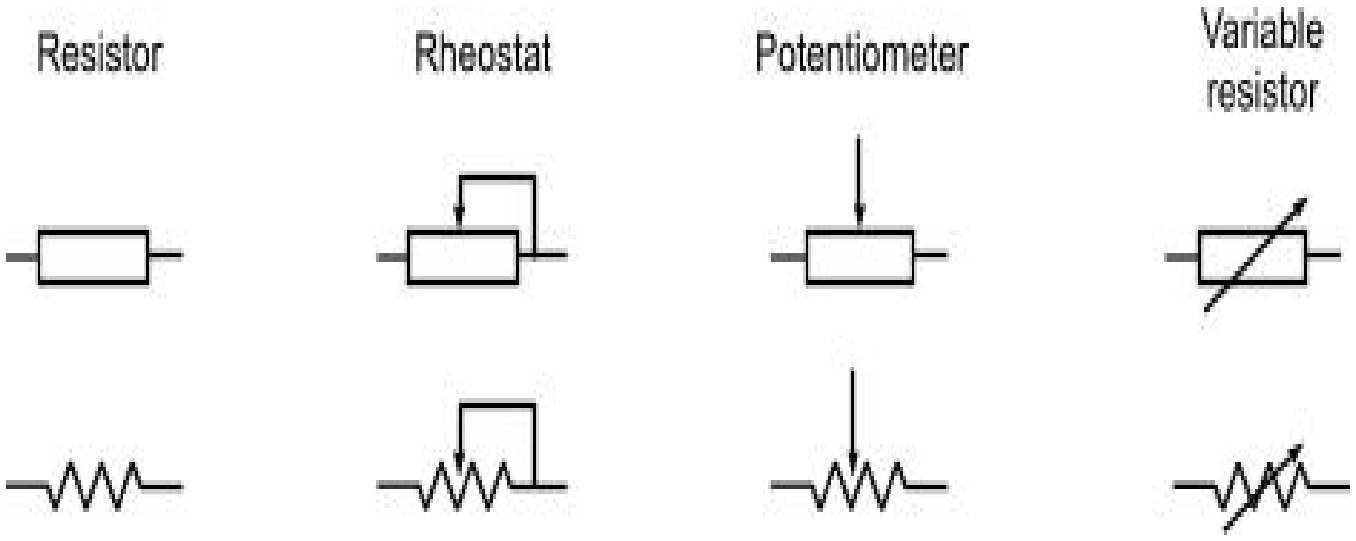
  Reduce current flow
  Adjust signal levels
  Divide voltages
  Bias active elements
  Terminate transmission lines
  Act as a passive load

2.3 Resistors

2 Types of Resistor

1. Fixed-resistance that only change slightly with temperature, time or operating voltage.

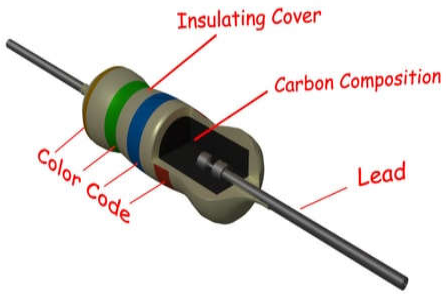
2. Variable-used to adjust circuit elements or as sensing devices for heat, light, humidity, force, or chemical activity.



2.3 Resistors

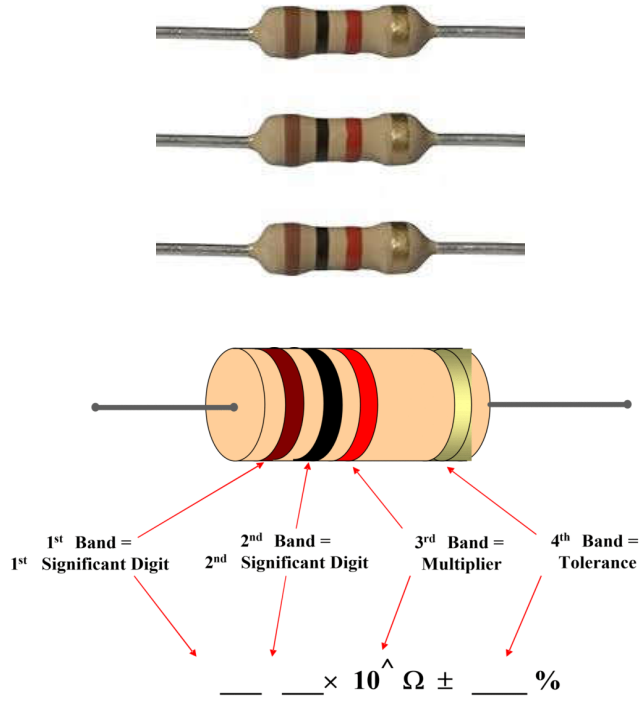
Types of Fixed Resistor

1. Carbon Composition



Color	Value	Multiplier	Tolerance
Black	0	x 10^0	±20%
Brown	1	x 10^1	± 1%
Red	2	x 10^2	± 2%
Orange	3	x 10^3	± 3%
Yellow	4	x 10^4	-0,+100%
Green	5	x 10^5	± 0.5%
Blue	6	x 10^6	± 0.25%
Violet	7	x 10^7	± 0.10%
Gray	8	x 10^8	±0.05%
White	9	x 10^9	± 10%
Gold	–	x 10^{-1}	± 5%
Silver	–	x 10^{-2}	± 10%
None	–	–	±20%

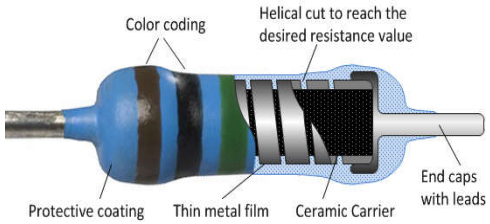
Color Code of Carbon Resistor



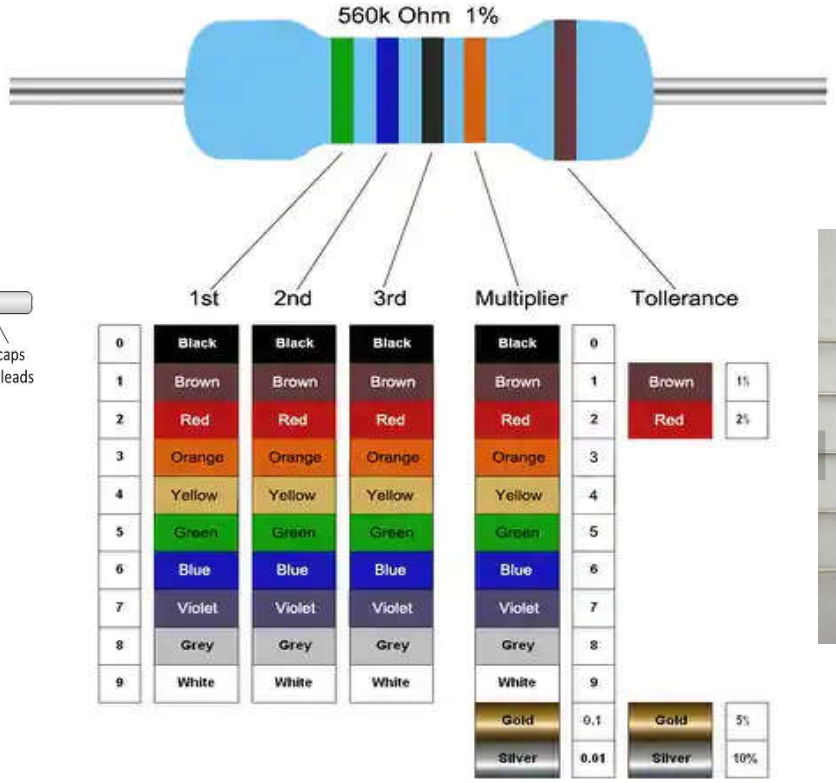
2.3 Resistors

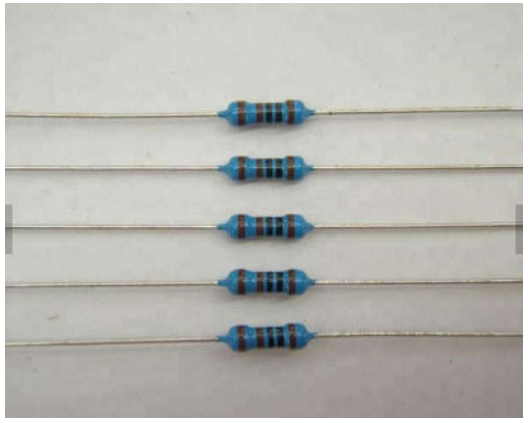
Types of Fixed Resistor

2. Metal Film



Find Tollerance Band (Usually Separated) and work from other side

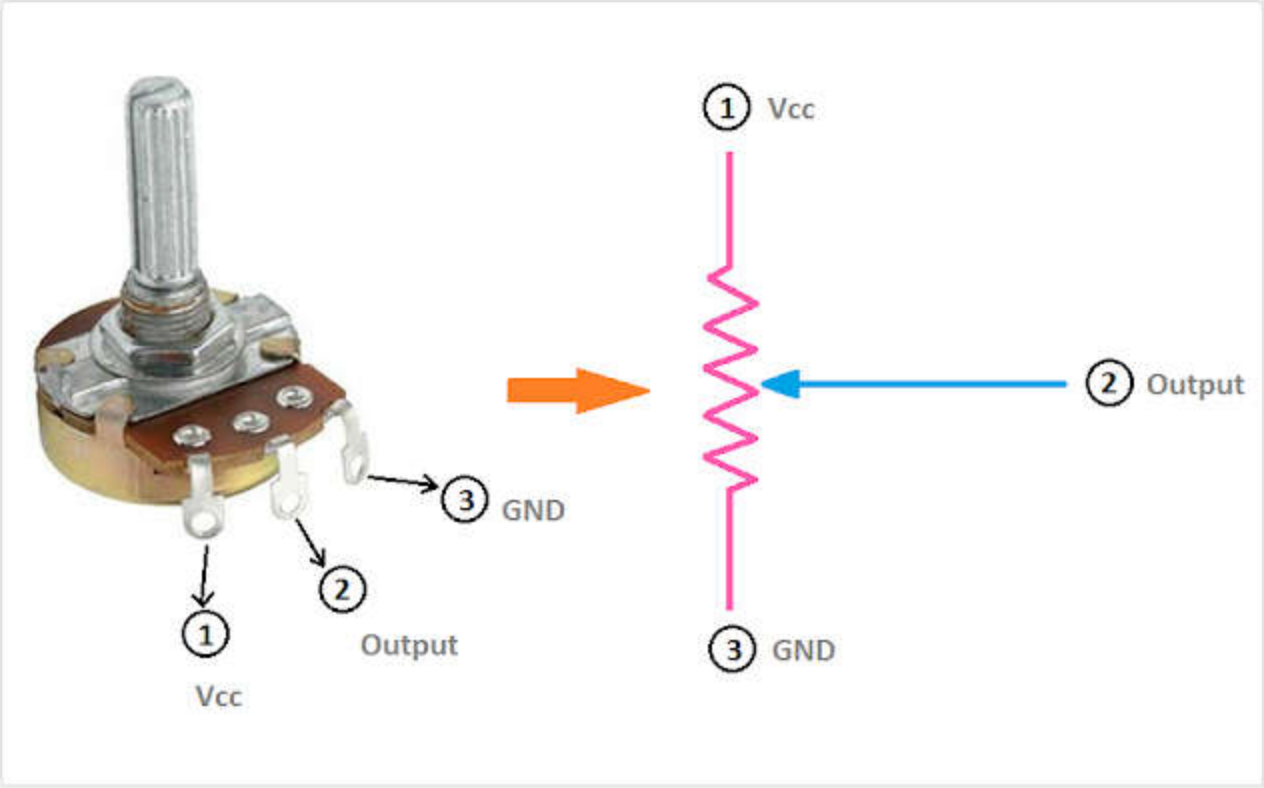




2.3 Resistors

Types of Variable Resistor

1. Potentiometer

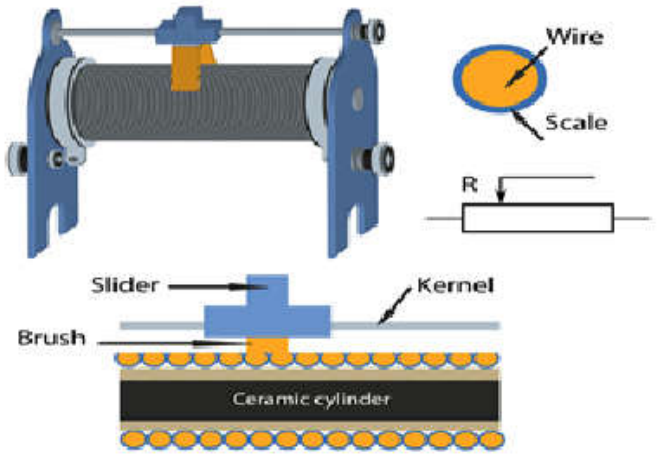


2.3 Resistors

Types of Variable Resistor

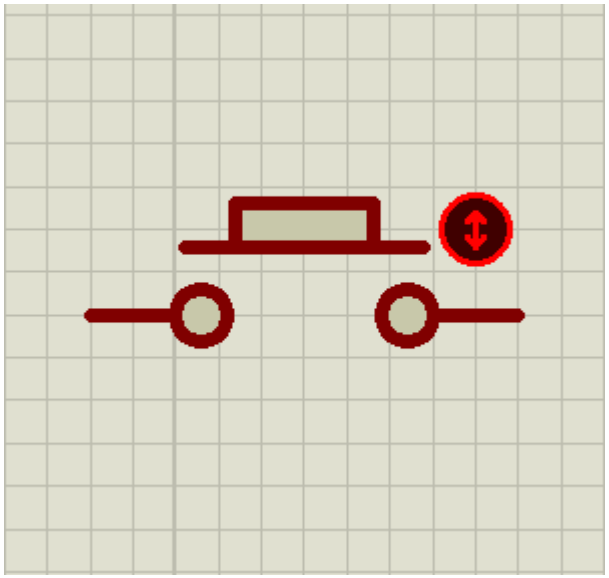
2. Rheostat

Practical Rheostats



2.4 Buttons: Pull-up and Pull-down

- Push Buttons

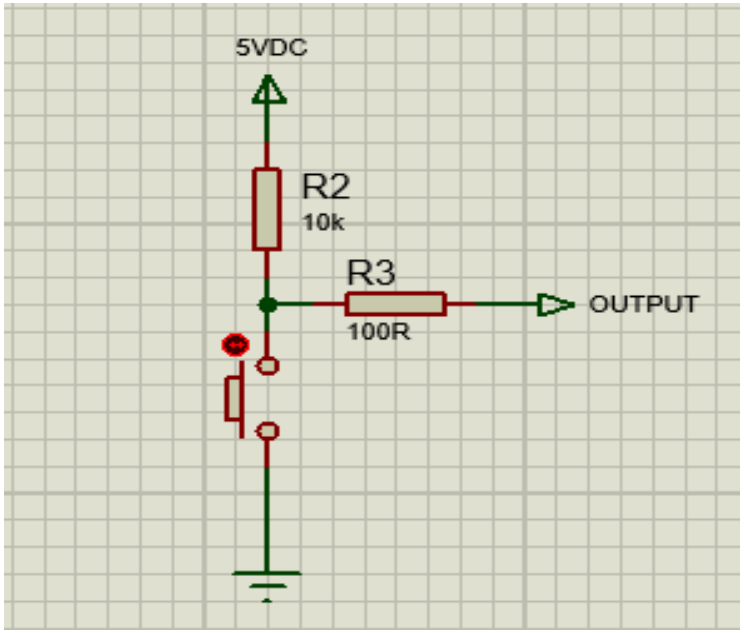


- Contact Bounce

Figure 1

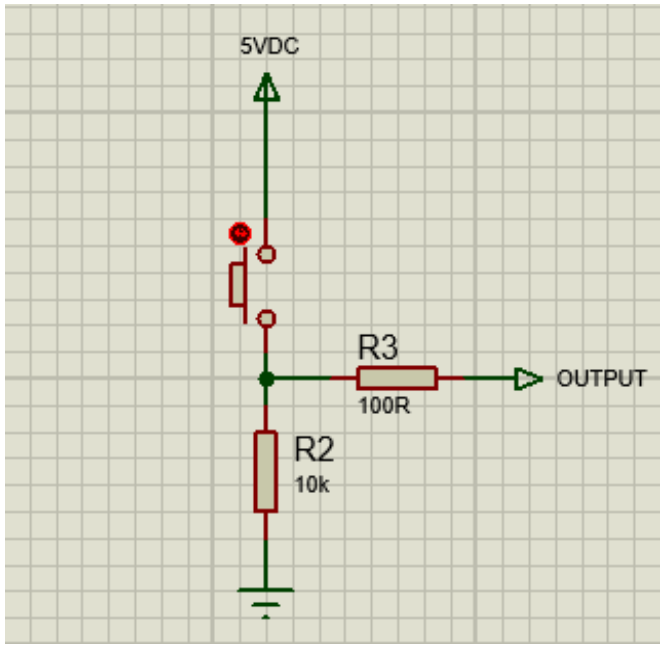
2.4 Buttons: Pull-up and Pull-down

- Pull-up
  - Active LOW Configuration



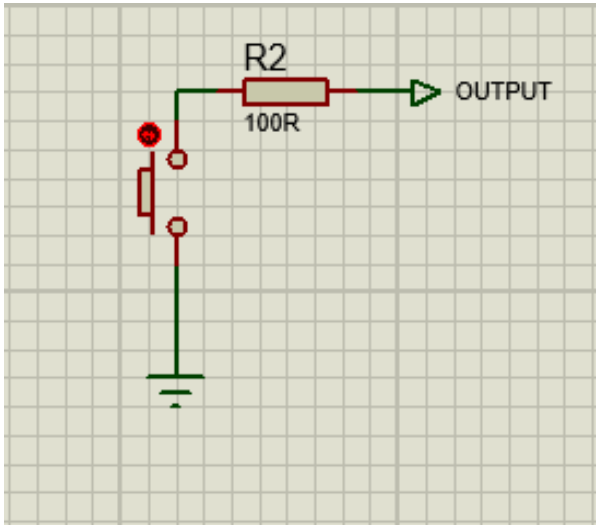
2.4 Buttons: Pull-up and Pull-down

- Pull-down
  - Active HIGH Configuration

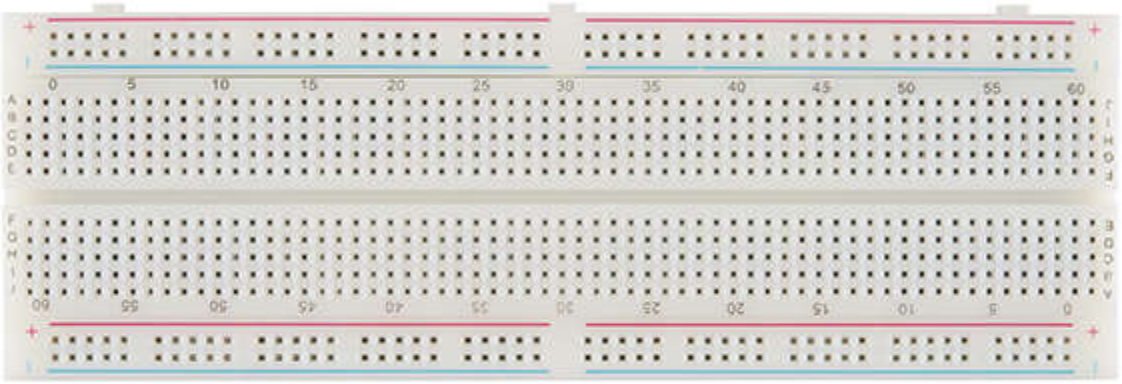


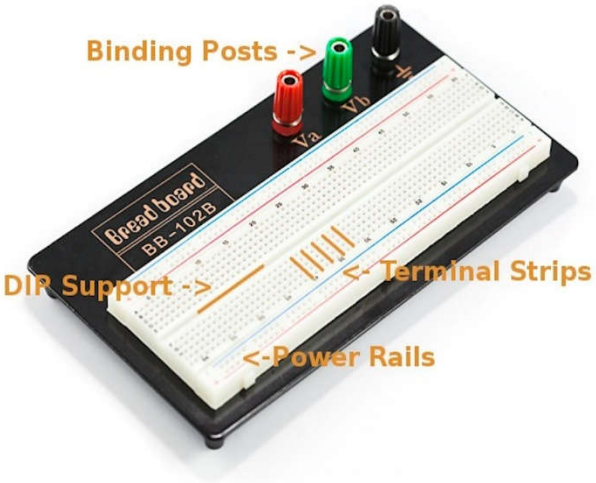
2.4 Buttons: Pull-up and Pull-down

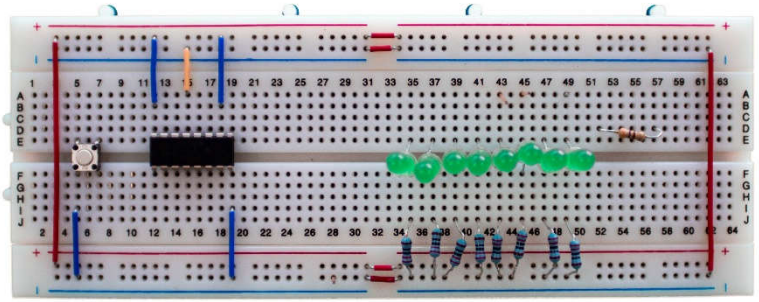
- Weak Pull-up
  - Active HIGH Configuration



2.5 Breadboard and Jumper Wires

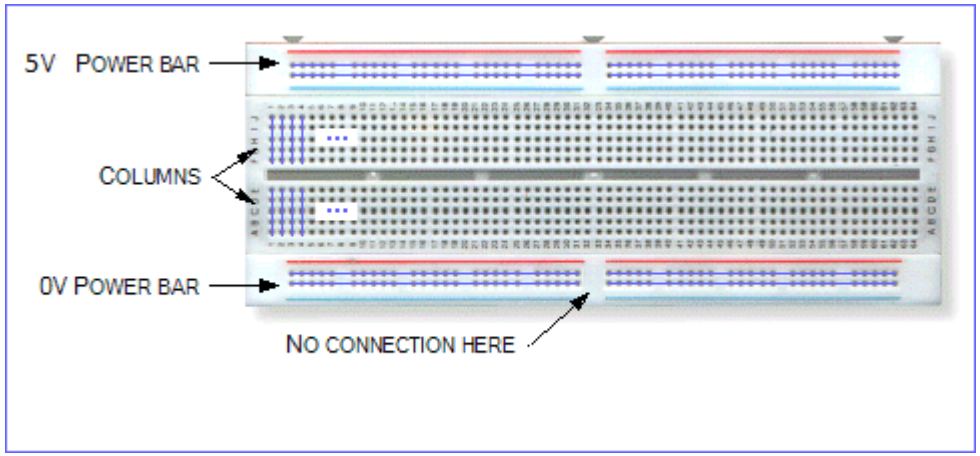


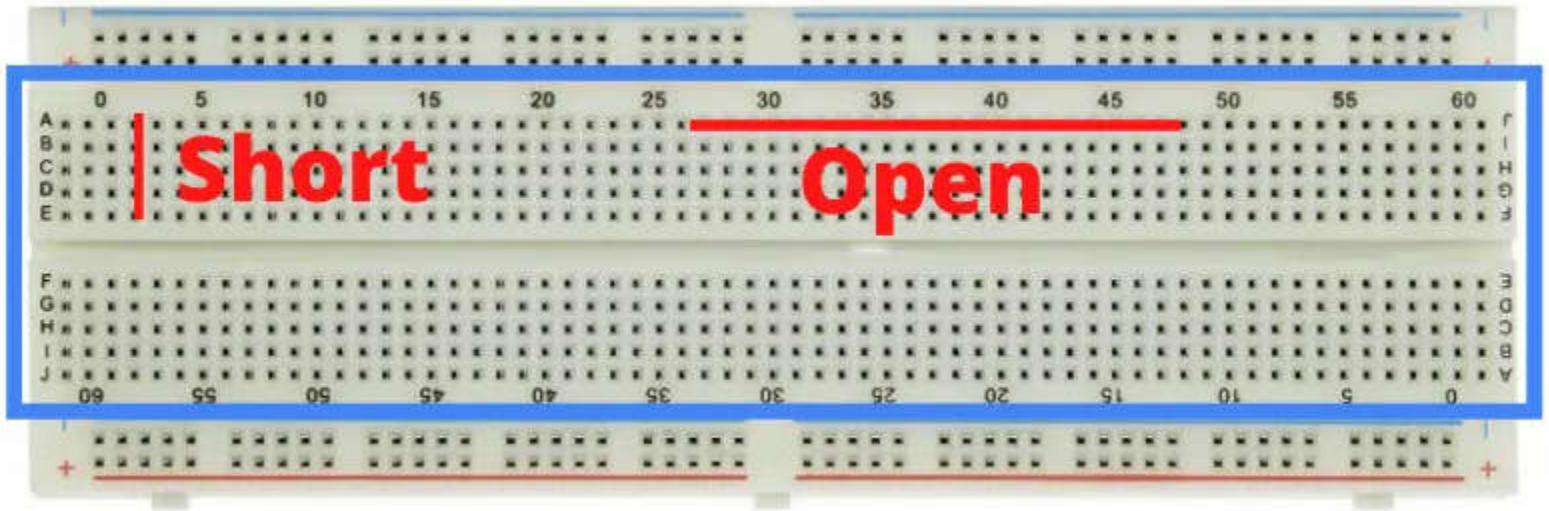




2.5 Breadboard and Jumper Wires

Breadboard Connection





2.6 Blinking LED Project: "Hello World" of Embedded Systems

- **Blinking LED- Hello World of Embedded Systems**

```
//Example 2.6 - BlinkingLED.ino
//Author: Rally Uminga
//Description: Blinking LED in pin 13 of Arduino Uno

int LED13 = 13;                    //D13 has built-in LED in Arduino Uno

void setup()              //All initialization put here
{

pinMode(LED13, OUTPUT);         //Initialize pin 13 as output

}

void loop()               //The main program that is repeated all over again
{

digitalWrite(LED13, HIGH);          //Writing the pin 13 into output high
delay(1000);              //Delay 1000 millisecond of Arduino delay function
digitalWrite(LED13, LOW);           //Writing the pin 13 into output low
delay(1000);              //Delay 1000 millisecond of Arduino delay function

}
```


2.7 Code Explanation

- **Comment**

C++ Style comment- double slash for single line comment

```
//Example 1: BlinkingLED.ino
//Author: Rally Uminga
//Description: Blinking LED in pin 13 of Arduino Uno
```

or

C Style comment-multiple statements

```
/*Example 1: BlinkingLED.ino
Author: Rally Uminga
Description: Blinking LED in pin 13 of Arduino Uno*/
```

2.7 Code Explanation

- **Variables**

```
int LED13 = 13;//D13 has built-in LED in Arduino Uno
```

int is a signed 16 bit integer data type from -32768 to 32767 values.

LED13 is a variable name replacing the pinout 13 of the Arduino Uno

2.7 Code Explanation

- **Arduino Data Types**

Data type	RAM	Number Range
void keyword	N/A	N/A
boolean	1 byte	0 to 1 (True or False)
byte	1 byte	0 to 255
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 65,535
word	2 byte	0 to 65,535
long	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long	4 byte	0 to 4,294,967,295
float	4 byte	-3.4028235E+38 to 3.4028235E+38
double	4 byte	-3.4028235E+38 to 3.4028235E+38
string	1 byte + x	Arrays of chars
array	1 byte + x	Collection of variables

2.7 Code Explanation

- **Setup function prototype**

```
void setup()            //All initialization put here
{

pinMode(LED13, OUTPUT);//Initialize pin 13 as output

}
```

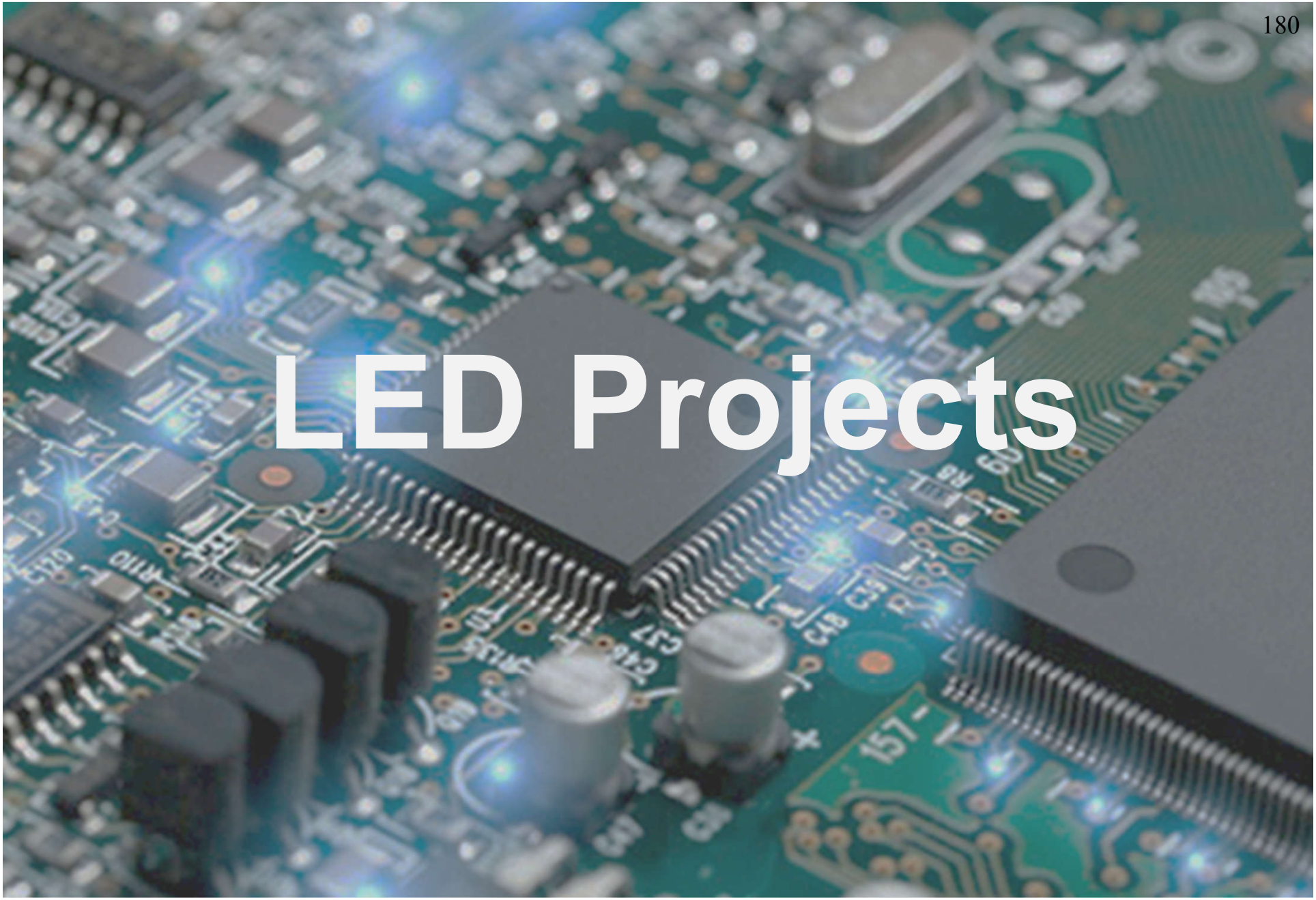
All initialization will be made here at void setup(). The code made here will only run once in entire program cycle.

2.7 Code Explanation

- **Loop function prototype**

```
void loop()    //The main program that is repeated all over
again
{

digitalWrite(LED13, HIGH);              //Writing the pin 13
into output high
delay(1000);   //Delay 1000 millisecond of Arduino delay
function
digitalWrite(LED13, LOW);                //Writing the pin 13
into output low
delay(1000);   //Delay 1000 millisecond of Arduino delay
function

}
```

The code will be executed repeatedly forever as part of a Arduino code or sketch to blink the LED on and off for 1 second.



LED Projects

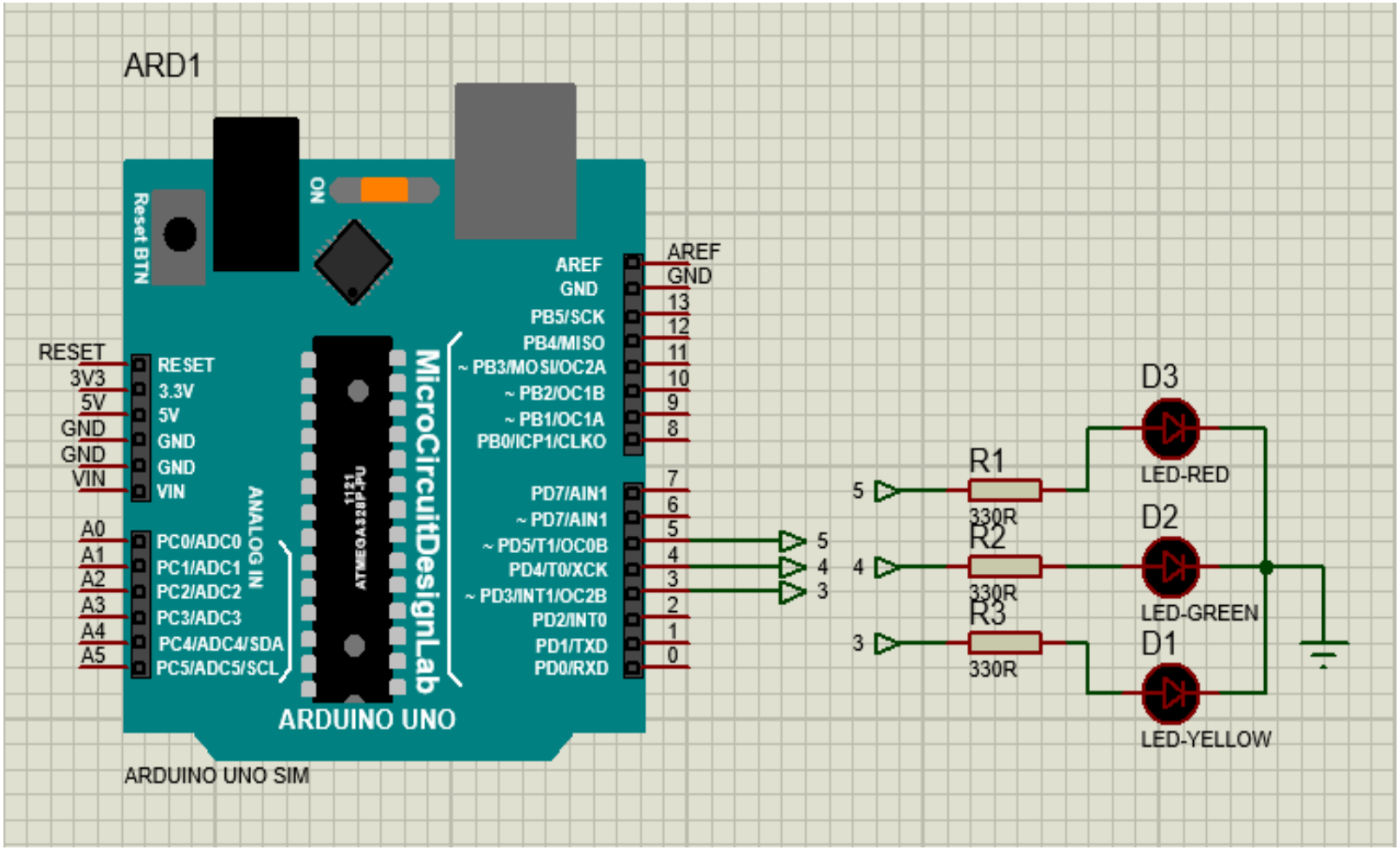
LED Projects

Contents

3.0 Objectives

- In this tutorial, you will learn:
  - To understand how to implement Traffic LEDs simulation in Arduino.
  - To know how to make Scrolling LEDs and potentiometer.
  - To become familiar using Pulsating LEDs with PWM output.
  - To understand how to make Mood Lamp using PWM output.
  - To know how to use LED Candle Effect using PWM.
  - To be able to start LED Effects using Arduino Uno.
  - To appreciate how use LED Effects.
  - To understand how use LED Matrix 8X8 and 74HC595.
  - To become familiar how use LED Matrix 8X8 and MAX7219.
  - To know how use LED Matrix 8X8 and MAX7219 in Ping Pong Game.

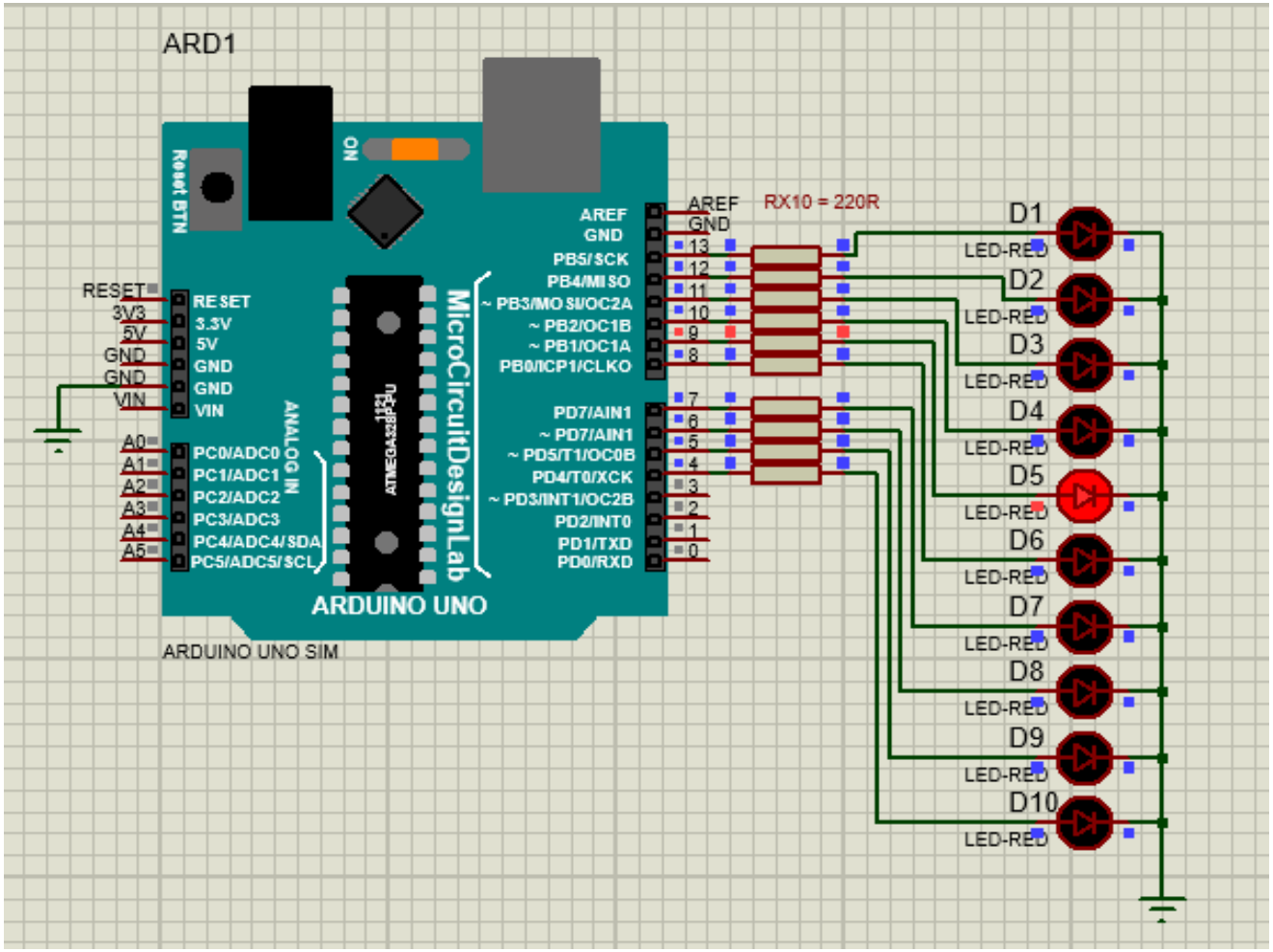
3.1 Traffic Light LEDs



3.1 Traffic Light LEDs

```
// Example 3.1 - TrafficLightsLED.ino

int LED_DELAY = 10000;              //Traffic Lights Delay
int RED_LED = 5;                    //Set RED_LED to pin 5
int YELLOW_LED = 4;                 //Set RED_LED to pin 4
int GREEN_LED = 3;                  //Set RED_LED to pin 3

void setup()
{

pinMode(RED_LED, OUTPUT);           //Initilalize RED_LED to output
pinMode(YELLOW_LED, OUTPUT);        //Initialize YELLOW_LED to output
pinMode(GREEN_LED, OUTPUT);         //Initialize GREEN_LED to output

}

void loop()
{

digitalWrite(RED_LED, HIGH);         // RED_LED ON
delay(LED_DELAY);                    // 10 seconds delay
digitalWrite(YELLOW_LED, HIGH);      // YELLOW_LED ON
delay(2000);                         // 2 seconds delay
digitalWrite(GREEN_LED, HIGH);       // GREEN_LED ON
digitalWrite(RED_LED, LOW);          // RED_LED OFF
digitalWrite(YELLOW_LED, LOW);       // YELLOW_LED OFF
delay(LED_DELAY);                    // 10 seconds delay
digitalWrite(YELLOW_LED, HIGH);      // YELLOW_LED ON
digitalWrite(GREEN_LED, LOW);        // GREEN_LED OFF
delay(2000);                         // 2 seconds delay
digitalWrite(YELLOW_LED, LOW);       // YELLOW_LED OFF
                                     // loop forever
}
```

3.2 Scrolling LEDs



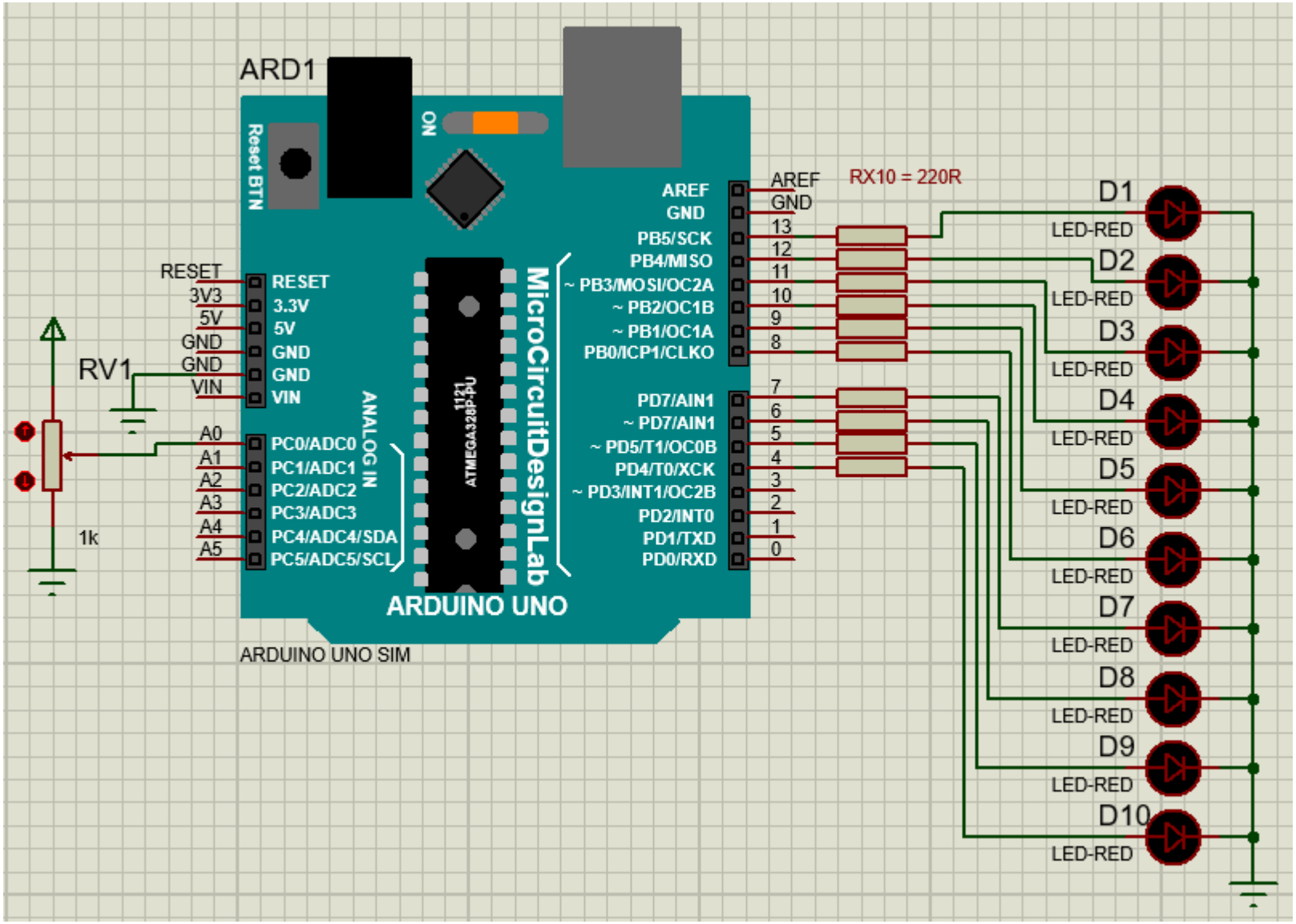
3.2 Scrolling LEDs

```
// Example 3.2 - ScrollingLED.ino
byte LED_Pins[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};   // LED pins array
int LED_Delay(65);                                      // Delay between LED changes
int LED_direction = 1;                                  // Direction flag
int current_LED = 0;                                    // Initialize LED direction
unsigned long elapsed_Time;                             // Elapsed time value for millis

void setup()
{
    for (int x=0; x<10; x++)
    {                                                   // Loop to set all pins to output
        pinMode(LED_Pins[x], OUTPUT);
    }
        elapsed_Time = millis();                        // Equate elapsedTime to millis()
}

void loop()
{
    if ((millis() - elapsed_Time) > LED_Delay)
    {                                                   // LED_Delay ms since last change
      change_LED();
      elapsed_Time = millis();
    }
}
```

3.2 Scrolling LEDs

```
void change_LED() {
    for (int x=0; x<10; x++) {                          // Turn OFF all
LED's
       digitalWrite(LED_Pins[x], LOW);
    }
    digitalWrite(LED_Pins[current_LED], HIGH);           // Turn ON the
current LED
    current_LED += LED_direction;                       // Increment by the
direction value

                                                        // Change direction
if we reach the end
    if (current_LED == 9) {LED_direction = -1;}
    if (current_LED == 0) {LED_direction = 1;}
}
```

3.3 Scrolling LEDs with Potentiometer



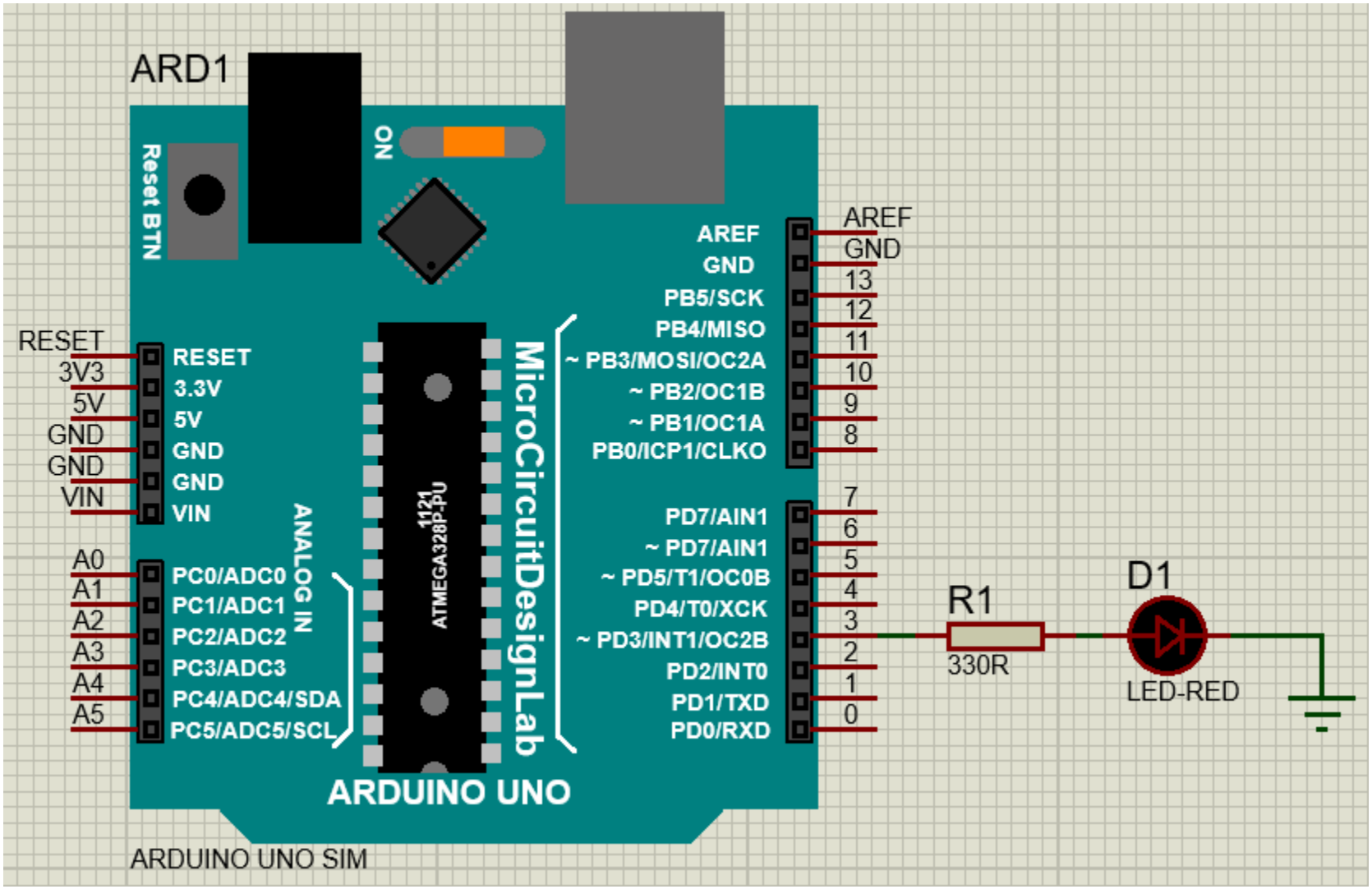
3.3 Scrolling LEDs with Potentiometer

```
// Example 3.3 - ScrollingLED_POT.ino
byte LED_Pins[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 13};   // LED pins array
int LED_Delay;                                          // Delay between LED changes
int LED_direction = 1;                                  // Direction flag
int current_LED = 0;                                    // Initialize LED direction
int pot_Pin = A1;                                       // A1 is input pin for the
potentiometer
unsigned long elapsed_Time;                             // Elapsed time value for millis

void setup()
{
    for (int x=0; x<10; x++)
    {                                                   // Loop to set all pins to output
        pinMode(LED_Pins[x], OUTPUT);
    }
        elapsed_Time = millis();                        // Equate elapsedTime to millis()
}

void loop()
{
    LED_Delay = analogRead(pot_Pin);                     // Read the value from the pot in A1

    if ((millis() - elapsed_Time) > LED_Delay)
    {                                                   // LED_Delay ms since last change
      change_LED();
      elapsed_Time = millis();
    }
}
```

3.3 Scrolling LEDs with Potentiometer

```
void change_LED()
{


    for (int x=0; x<10; x++) {                          // Turn OFF all LED's
       digitalWrite(LED_Pins[x], LOW);
    }
    digitalWrite(LED_Pins[current_LED], HIGH);           // Turn ON the current LED
    current_LED += LED_direction;                       // Increment by the direction value
                                                        // Change direction if we reach the end
    if (current_LED == 9) {LED_direction = -1;}
    if (current_LED == 0) {LED_direction = 1;}
}
```

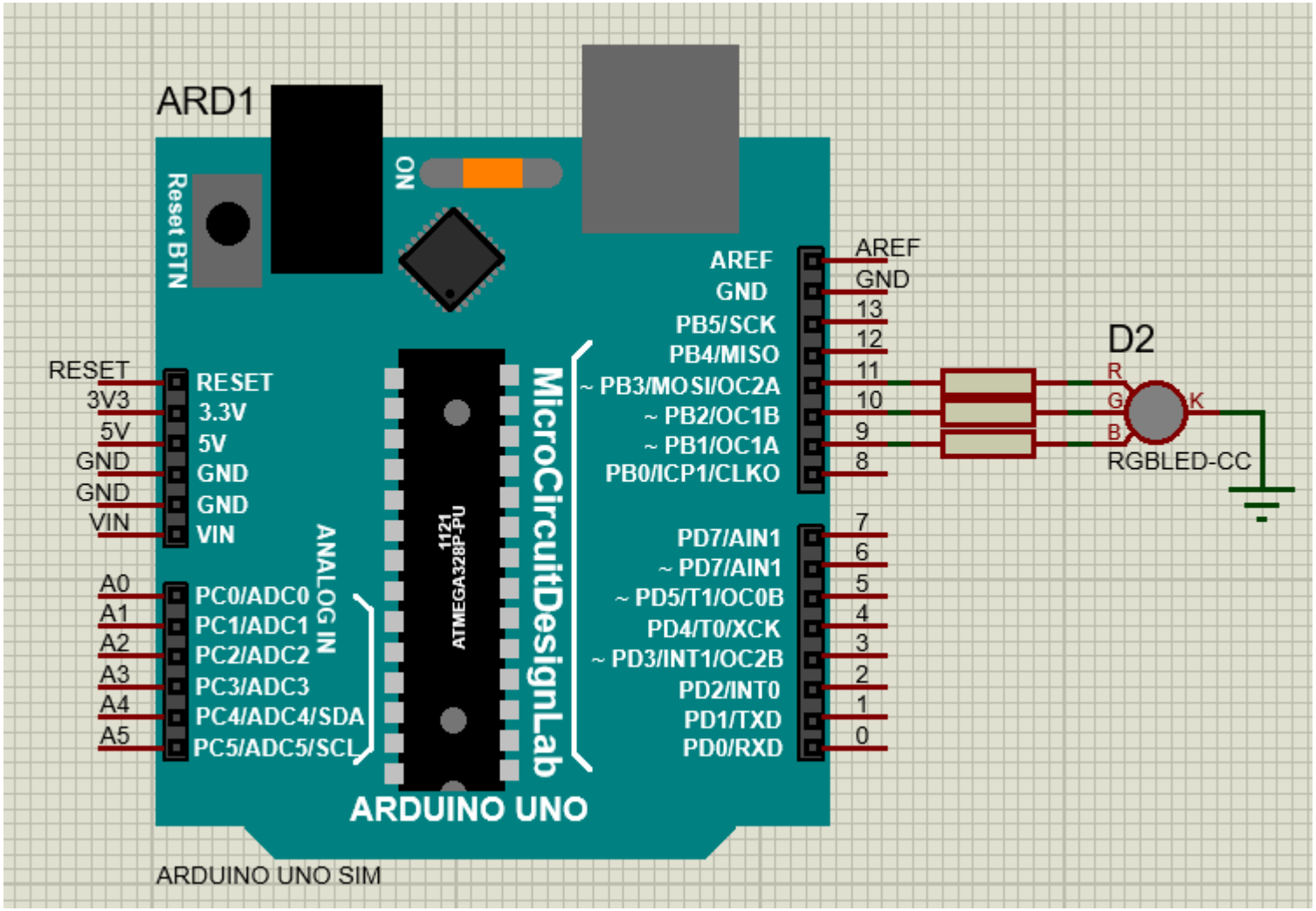
3.4 Pulsating LED



3.4 Pulsating LED

```
// Example 3.4 - PulsatingLED.ino
int LED_Pin = 3;                          // Pin 3 PWM output
float sine_Val;                           // Sine values
int LED_Val;                              // LED vales


void setup()
{

    pinMode(LED_Pin, OUTPUT);             // Initialize LED_Pin to output

}

void loop()
{
    for (int x=0; x<180; x++)             // Iterate until 180 times
    {

    sine_Val = (sin(x*(3.1416/180)));     // Degrees to Radians for sine value
    LED_Val = int(sine_Val*255);          // 8-bit PWM value
    analogWrite(LED_Pin, LED_Val);        // PWM Output
    delay(10);                            // Delay 10ms

    }
}
```

3.5 Mood Lamp


ARD1
Reset BTN
ON
RESET
3V3
5V
GND
GND
VIN
RESET
3.3V
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ATMEGA328P-PU
MicroCircuitDesignLab
AREF
GND
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
D2
R
G
B
K
RGBLED-CC
ARDUINO UNO
ARDUINO UNO SIM

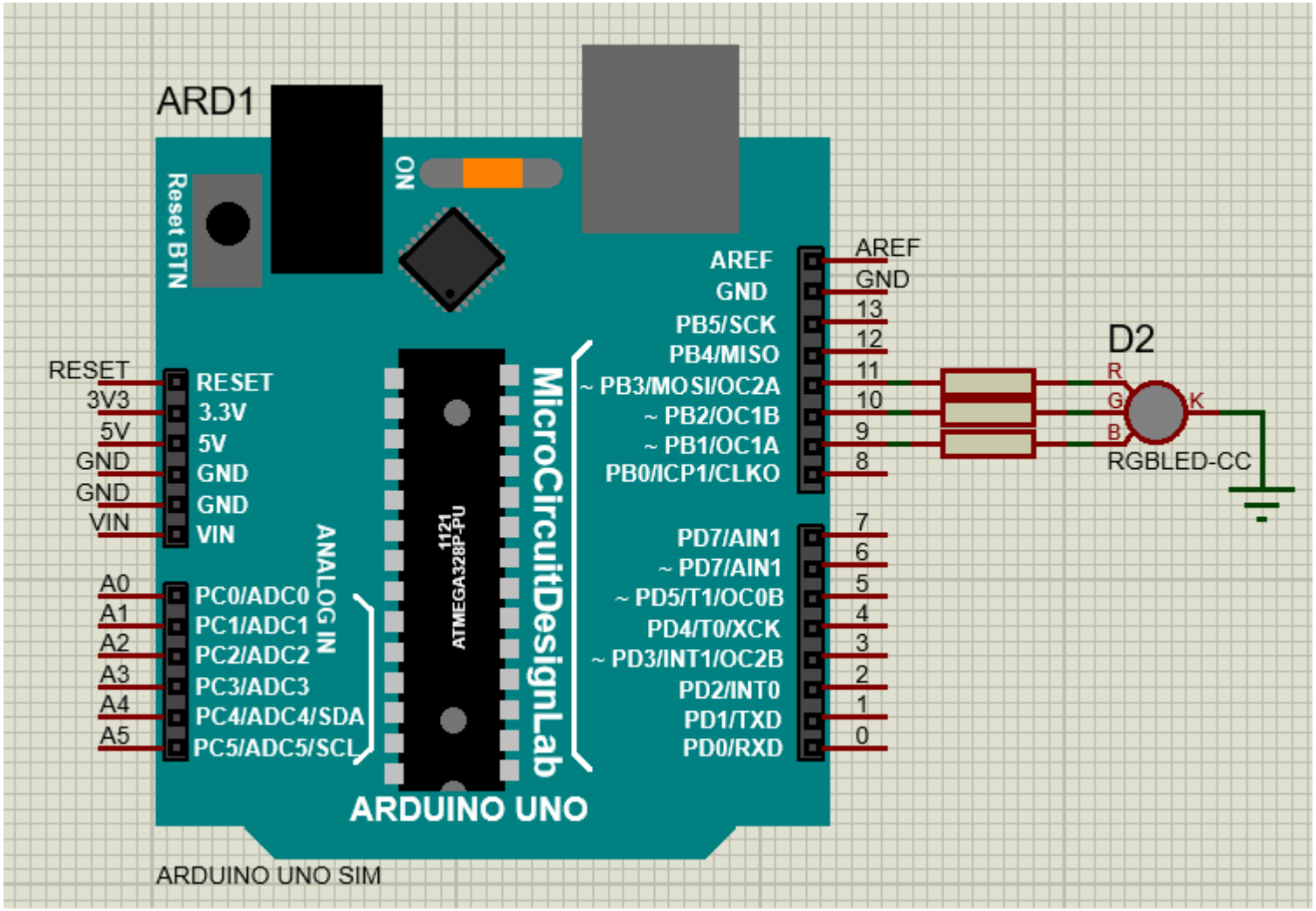
3.5 Mood Lamp

```
// Example 3.5 - MoodLampLED.ino
float RGB1[3];
float RGB2[3];
float INC[3];
int red, green, blue;
int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;

void setup()
{
randomSeed(analogRead(0));
RGB1[0] = 0;
RGB1[1] = 0;
RGB1[2] = 0;
RGB2[0] = random(256);
RGB2[1] = random(256);
RGB2[2] = random(256);
}
```

3.5 Mood Lamp

```
void loop()
{
randomSeed(analogRead(0));
for (int x=0; x<3; x++)
{
INC[x] = (RGB1[x] - RGB2[x]) / 256;
}

for (int x=0; x<256; x++)
{
red = int(RGB1[0]);
green = int(RGB1[1]);
blue = int(RGB1[2]);
analogWrite (RedPin, red);
analogWrite (GreenPin, green);
analogWrite (BluePin, blue);
delay(100);
RGB1[0] -= INC[0];
RGB1[1] -= INC[1];
RGB1[2] -= INC[2];
}

for (int x=0; x<3; x++)
{
RGB2[x] = random(556)-300;
RGB2[x] = constrain(RGB2[x], 0, 255);
delay(1000);
}
}
```

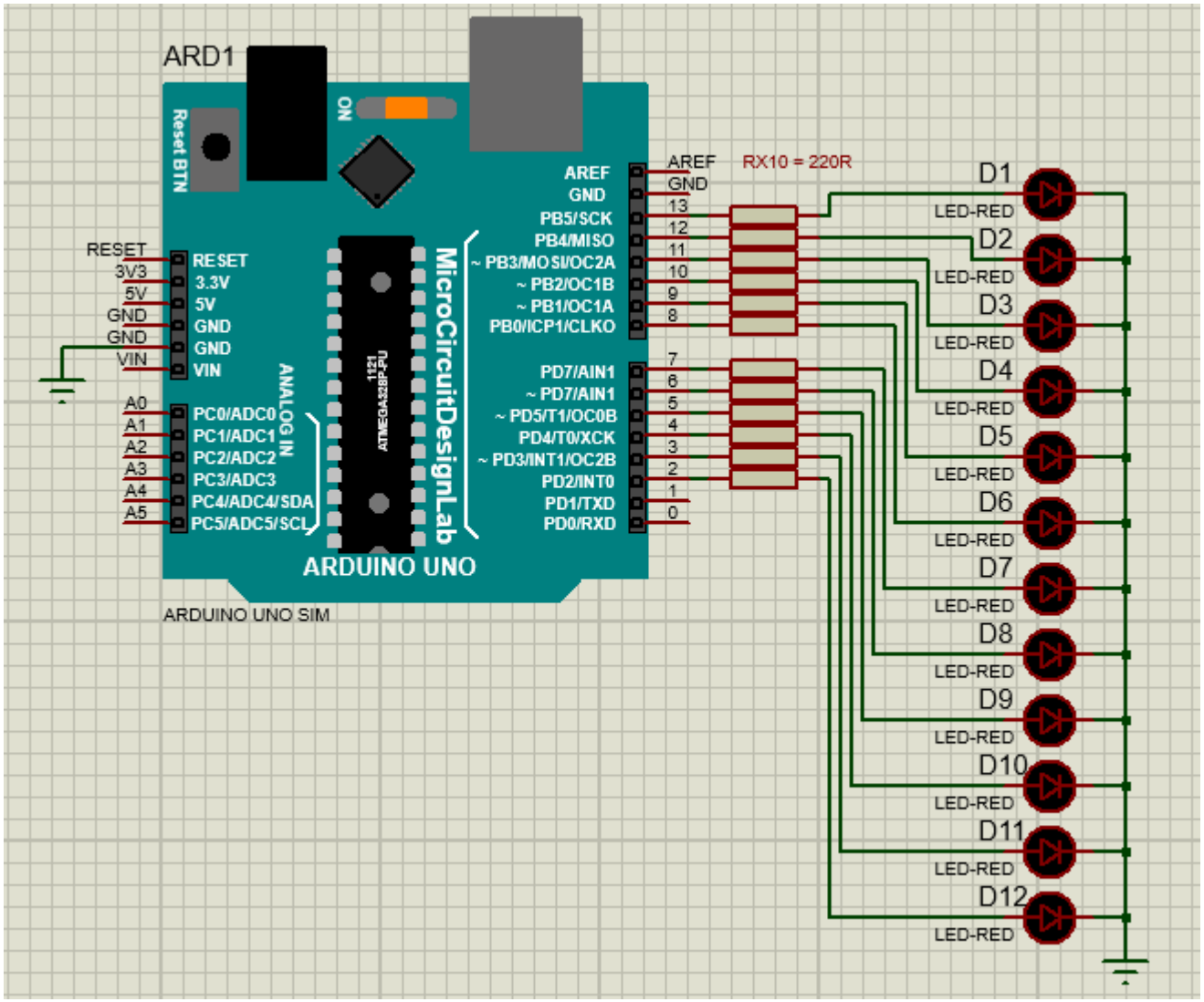
3.6 LED Candle Effect



3.6 LED Candle Effect

```
// Example 3.6 - LEDCandleEffect.ino
int LED_Pin1 = 9;                          // LED PWM1
int LED_Pin2 = 10;                         // LED PWM2
int LED_Pin3 = 11;                         // LED PWM3

void setup()
{
pinMode(LED_Pin1, OUTPUT);                 // Initialize Pin 9 to Output
pinMode(LED_Pin2, OUTPUT);                 // Initialize Pin 9 to Output
pinMode(LED_Pin3, OUTPUT);                 // Initialize Pin 9 to Output
}

void loop()
{
analogWrite(LED_Pin1, random(120)+135);    // Randomize Pin 9 PWM
analogWrite(LED_Pin2, random(120)+135);    // Randomize Pin 10 PWM
analogWrite(LED_Pin3, random(120)+135);    // Randomize Pin 11 PWM
delay(random(100));                        // Randomize delay in ms
}
```

3.7 LED Effects



3.7 LED Effects

```
// Example 3.7 - LED_Effects.ino
int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;
int led6 = 7;
int led7 = 8;
int led8 = 9;
int led9 = 10;
int led10 = 11;
int led11 = 12;
int led12 = 13;

void setup() {
  // put your setup code here, to run once:
 //initialize digital pin as output
 pinMode(led1, OUTPUT);
 pinMode(led2, OUTPUT);
 pinMode(led3, OUTPUT);
 pinMode(led4, OUTPUT);
 pinMode(led5, OUTPUT);
 pinMode(led6, OUTPUT);
 pinMode(led7, OUTPUT);
 pinMode(led8, OUTPUT);
 pinMode(led9, OUTPUT);
 pinMode(led10, OUTPUT);
 pinMode(led11, OUTPUT);
 pinMode(led12, OUTPUT);
}
```

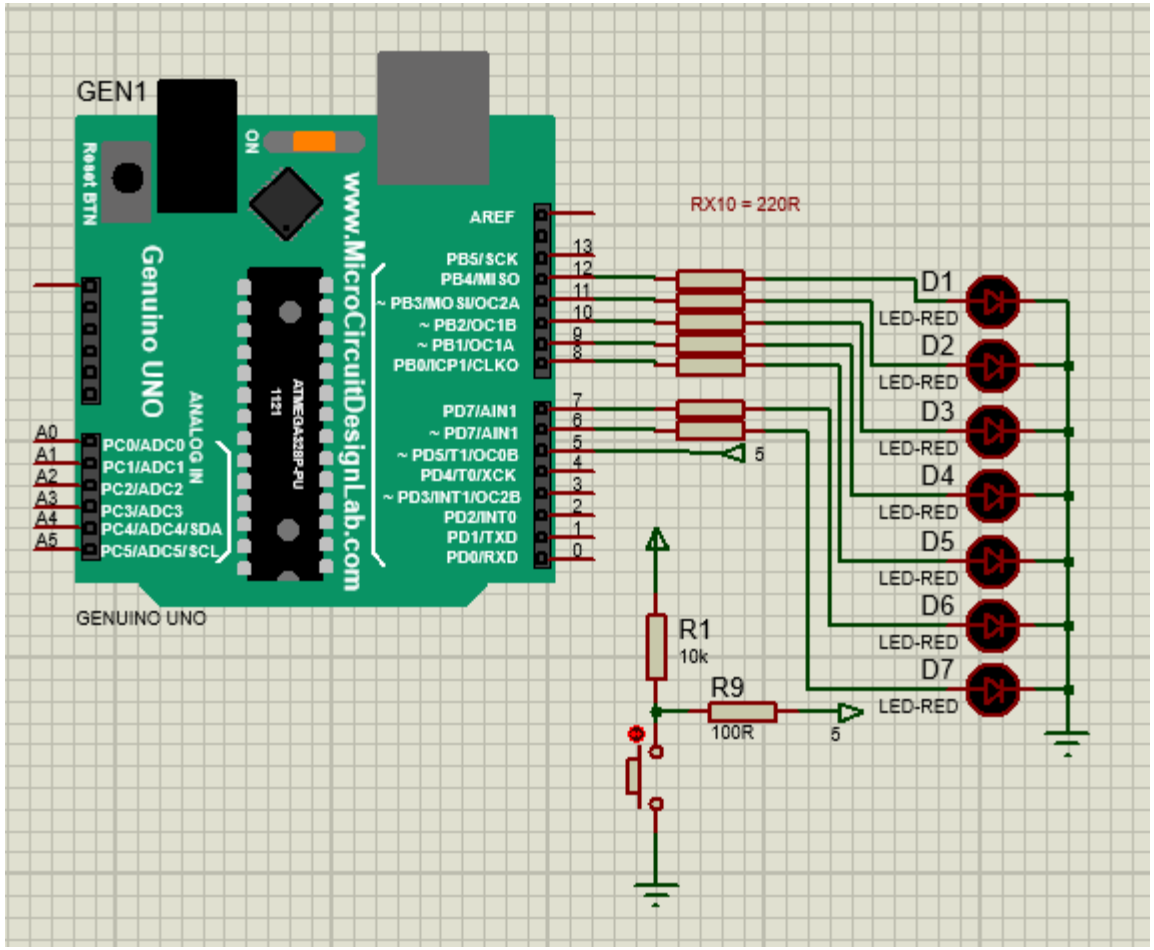
3.7 LED Effects

```
void loop()
{
 digitalWrite(led1, HIGH);
 delay(100);
 digitalWrite(led2, HIGH);
 delay(100);
 digitalWrite(led3, HIGH);
 delay(100);
 digitalWrite(led4, HIGH);
 delay(100);
 digitalWrite(led5, HIGH);
 delay(100);
 digitalWrite(led6, HIGH);
 delay(100);
 digitalWrite(led7, HIGH);
 delay(100);
 digitalWrite(led8, HIGH);
 delay(100);
 digitalWrite(led9, HIGH);
 delay(100);
 digitalWrite(led10, HIGH);
 delay(100);
 digitalWrite(led11, HIGH);
 delay(100);
 digitalWrite(led12, HIGH);
 delay(100);
```

3.7 LED Effects

```
 digitalWrite(led1, LOW);
 delay(100);
 digitalWrite(led2, LOW);
 delay(100);
 digitalWrite(led3, LOW);
 delay(100);
 digitalWrite(led4, LOW);
 delay(100);
 digitalWrite(led5, LOW);
 delay(100);
 digitalWrite(led6, LOW);
 delay(100);
 digitalWrite(led7, LOW);
 delay(100);
 digitalWrite(led8, LOW);
 delay(100);
 digitalWrite(led9, LOW);
 delay(100);
 digitalWrite(led10, LOW);
 delay(100);
 digitalWrite(led11, LOW);
 delay(100);
 digitalWrite(led12, LOW);
 delay(100);
}
```

3.8 Button and LED Effects



3.8 Button and LED Effects

```
// Example 3.8 - LEDEffectsButton.ino
int LED_12 = 12;
int LED_11 = 11;
int LED_10 = 10;
int LED_9 = 9;
int LED_8 = 8;
int LED_7 = 7;
int LED_6 = 6;

int buttonPin_5 = 5;                 //the number of the pushbutton pin

int delay_time = 100;                // time delay

int p = 0;                           // variable for pattern

bool buttonState = 1;                // variable for reading the pushbutton status

void setup()
{
  pinMode(LED_12, OUTPUT);
  pinMode(LED_11, OUTPUT);
  pinMode(LED_10, OUTPUT);
  pinMode(LED_9, OUTPUT);
  pinMode(LED_8, OUTPUT);
  pinMode(LED_7, OUTPUT);
  pinMode(LED_6, OUTPUT);

  pinMode(buttonPin_5, INPUT);
}
```

3.8 Button and LED Effects

```
void loop()
{

 buttonState = digitalRead(buttonPin_5);

 if (buttonState == LOW)

    {
      p++;
      delay(50);
    }

  if(p==1)
  {
 digitalWrite(LED_12,1);
 digitalWrite(LED_11,0);
 digitalWrite(LED_10,0);
 digitalWrite(LED_9,0);
 digitalWrite(LED_8,0);
 digitalWrite(LED_7,0);
 digitalWrite(LED_6,0);  //1
 delay(delay_time);

 digitalWrite(LED_12,0);
 digitalWrite(LED_11,1);
 digitalWrite(LED_10,0);
 digitalWrite(LED_9,0);
 digitalWrite(LED_8,0);
 digitalWrite(LED_7,0);
 digitalWrite(LED_6,0);  //2
 delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //5
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //6
delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //7
delay(delay_time);
 }
 if(p==2)
 {
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //7
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //6
delay(delay_time);
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //5
delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);


digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //2
delay(delay_time);

digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
```

3.8 Button and LED Effects

```
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //1
delay(delay_time);

 }
   if(p==3)
 {
digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //1
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //2
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //6
delay(delay_time);

 digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //5
delay(delay_time);

 digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //2
delay(delay_time);
 }
   if(p==4)
 {
digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //1,7
delay(delay_time);
digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //2,6
delay(delay_time);
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3,5
delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

 }

     if(p==5)
 {

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3,5
delay(delay_time);
```

3.8 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //2,6
delay(delay_time);

digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //1,7
delay(delay_time);
 }
```

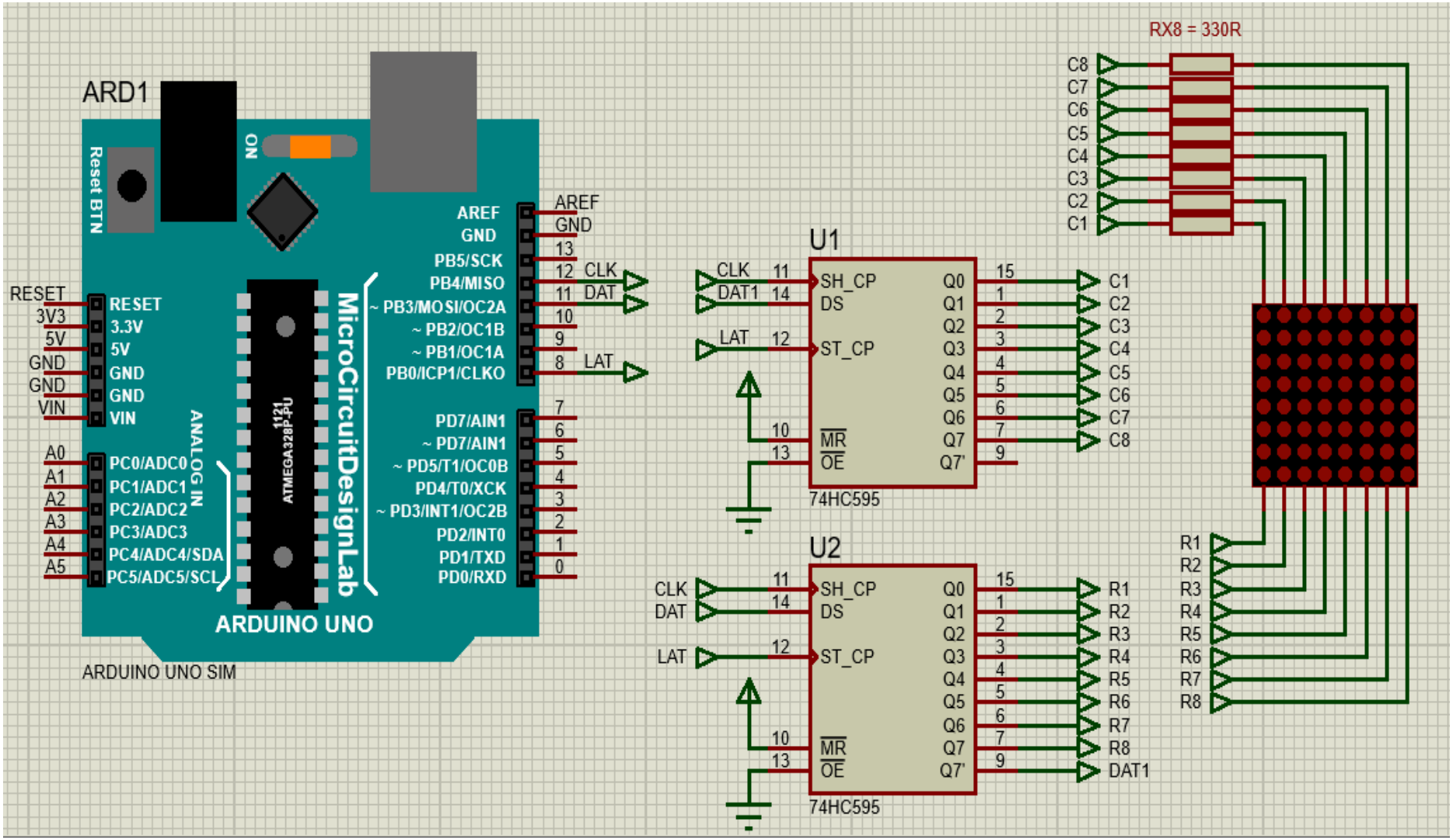
3.8 Button and LED Effects

```
 if(p==6)
{
 digitalWrite(LED_12,1);
 delay(delay_time);
 digitalWrite(LED_11,1);
 delay(delay_time);
 digitalWrite(LED_10,1);
 delay(delay_time);
 digitalWrite(LED_9,1);
 delay(delay_time);
 digitalWrite(LED_8,1);
 delay(delay_time);
 digitalWrite(LED_7,1);
 delay(delay_time);
 digitalWrite(LED_6,1);  //1,7
 delay(delay_time);
 digitalWrite(LED_6,0);  //1,7
 delay(delay_time);
 digitalWrite(LED_7,0);
 delay(delay_time);
 digitalWrite(LED_8,0);
 delay(delay_time);
 digitalWrite(LED_9,0);
 delay(delay_time);
 digitalWrite(LED_10,0);
 delay(delay_time);
 digitalWrite(LED_11,0);
 delay(delay_time);
 digitalWrite(LED_12,0);
 delay(delay_time);

}
```

3.8 Button and LED Effects

```
 if(p==7)
{
 digitalWrite(LED_12,0);
 digitalWrite(LED_11,0);
 digitalWrite(LED_10,0);
 digitalWrite(LED_9,0);
 digitalWrite(LED_8,0);
 digitalWrite(LED_7,0);
 digitalWrite(LED_6,0);  //1,7
  p=0;
}

}
```

3.9 LED Matrix 8X8 using 74HC595



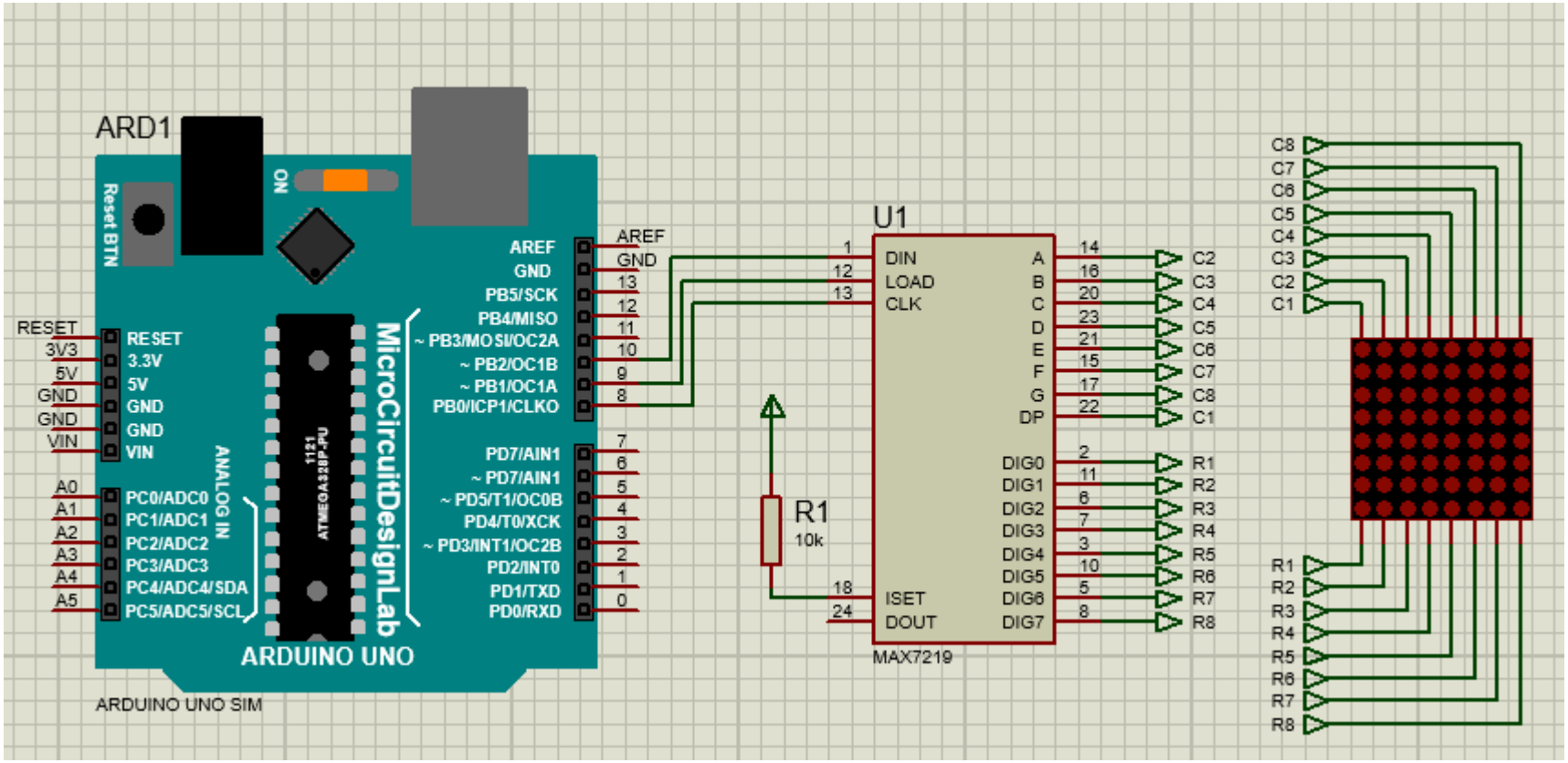
3.9 LED Matrix 8X8 using 74HC595

```
//Dot-Matrix Display
//The dot-matrix will display 0 to F circularly
//Example 3.9  - LEDMatrix8X8_74hc595.ino

const int latchPin = 8;                    //Pin connected to ST_CP of 74HC595
const int clockPin = 12;                   //Pin connected to SH_CP of 74HC595
const int dataPin = 11;                    //Pin connected to DS of 74HC595
int data[] = {
  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,/*" ",0*/
  0xFF,0xC1,0xBE,0xBE,0xBE,0xC1,0xFF,0xFF,/*"0",1*/
  0xFF,0xDF,0xDF,0x80,0xFF,0xFF,0xFF,0xFF,/*"1",2*/
  0xFF,0xDC,0xBC,0xBA,0xB6,0xCE,0xFF,0xFF,/*"2",3*/
  0xFF,0xDD,0xBE,0xB6,0xB6,0xC9,0xFF,0xFF,/*"3",4*/
  0xFB,0xF3,0xEB,0xDB,0x80,0xFB,0xFF,0xFF,/*"4",5*/
  0xFF,0x8D,0xAE,0xAE,0xAE,0xB1,0xFF,0xFF,/*"5",6*/
  0xFF,0xC1,0x96,0xB6,0xB6,0xD9,0xFF,0xFF,/*"6",7*/
  0xFF,0xBF,0xBC,0xB3,0xAF,0x9F,0xFF,0xFF,/*"7",8*/
  0xFF,0xC9,0xB6,0xB6,0xB6,0xC9,0xFF,0xFF,/*"8",9*/
  0xFF,0xCD,0xB6,0xB6,0xB4,0xC1,0xFF,0xFF,/*"9",10*/
  0xFC,0xF3,0xCB,0x9B,0xEB,0xF3,0xFC,0xFF,/*"A",11*/
  0xFF,0x80,0xB6,0xB6,0xB6,0xB6,0xC9,0xFF,/*"B",12*/
  0xFF,0xE3,0xDD,0xBE,0xBE,0xBE,0xBE,0xDD,/*"C",13*/
  0xFF,0x80,0xBE,0xBE,0xBE,0xBE,0xDD,0xE3,/*"D",14*/
  0xFF,0x80,0xB6,0xB6,0xB6,0xB6,0xBE,0xFF,/*"E",15*/
  0xFF,0x80,0xB7,0xB7,0xB7,0xB7,0xBF,0xFF,/*"F",16*/
  0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,/*" ",17*/
};
```

3.9 LED Matrix 8X8 using 74HC595

```
void setup ()
{
  //set pins to output
  pinMode(latchPin,OUTPUT);
  pinMode(clockPin,OUTPUT);
  pinMode(dataPin,OUTPUT);
}
void loop()
{
  for(int n = 0; n < 136; n++)
  {
    for(int t = 0;t < 10;t ++)                          //Show repeated 10 times
    {
      int dat = 0x01;
      for(int num = n; num < 8+n; num++)
      {
        shiftOut(dataPin,clockPin,MSBFIRST,~data[num]);     //control ROW of dot matrix
        shiftOut(dataPin,clockPin,MSBFIRST,~dat);           //control COL of dot matrix
        //return the latch pin high to signal chip that it
        //no longer needs to listen for information
        digitalWrite(latchPin,HIGH);                        //pull the latchPin to save the data
        //delay(1); //wait for a microsecond
        digitalWrite(latchPin,LOW);                         //ground latchPin and hold low for as
                                                           //long as you are transmitting
        dat = dat<<1;
        delay(1);                                           //wait for a microsecond
      }
    }
  }
}
```

3.10 LED Matrix 8X8 using MAX7219



3.10 LED Matrix 8X8 using MAX7219

```
//Example 3.10 - LEDMatrixMax7219.ino
//LED 8x8 Matrix Display DEMO using MAX7219

#include <LedControl.h>
int DIN = 10;
int CS =  9;
int CLK = 8;

LedControl lc = LedControl(DIN,CLK,CS,0);

void setup()
{
 lc.shutdown(0,false);
 lc.setIntensity(0,15);      //Adjust the brightness maximum is 15
 lc.clearDisplay(0);
}

void loop()
{
    //Facial Expression
    byte smile[8]=   {0x3C,0x42,0xA5,0x81,0xA5,0x99,0x42,0x3C};
    byte neutral[8]= {0x3C,0x42,0xA5,0x81,0xBD,0x81,0x42,0x3C};
    byte sad[8]=   {0x3C,0x42,0xA5,0x81,0x99,0xA5,0x42,0x3C};

    //Arrow
    byte arrow_up[8]= {0x18,0x3C,0x7E,0xFF,0x18,0x18,0x18,0x18};
    byte arrow_down[8]= {0x18,0x18,0x18,0x18,0xFF,0x7E,0x3C,0x18};

    //Alternate Pattern
    byte d1[8]= {0xAA,0x55,0xAA,0x55,0xAA,0x55,0xAA,0x55};
    byte d2[8]= {0x55,0xAA,0x55,0xAA,0x55,0xAA,0x55,0xAA};
```

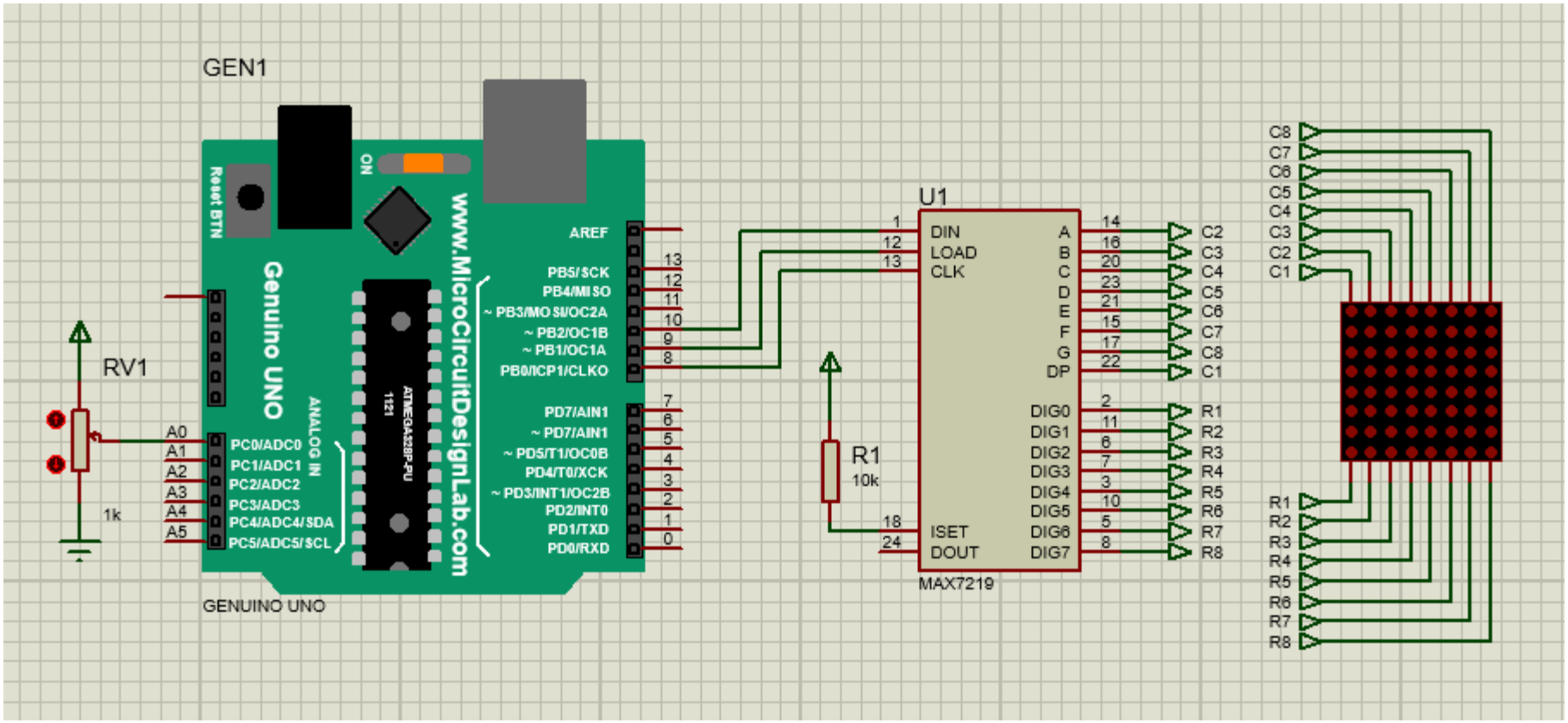
3.10 LED Matrix 8X8 using MAX7219

```
//Moving car
    byte b1[8]= {0x00,0x00,0x00,0x00,0x18,0x3C,0x18,0x3C};
    byte b2[8]= {0x00,0x00,0x00,0x18,0x3C,0x18,0x3C,0x00};
    byte b3[8]= {0x00,0x00,0x18,0x3C,0x18,0x3C,0x00,0x00};
    byte b4[8]= {0x00,0x18,0x3C,0x18,0x3C,0x00,0x00,0x00};
    byte b5[8]= {0x18,0x3C,0x18,0x3C,0x00,0x00,0x00,0x00};
    byte b6[8]= {0x3C,0x18,0x3C,0x00,0x00,0x00,0x00,0x18};
    byte b7[8]= {0x18,0x3C,0x00,0x00,0x00,0x00,0x18,0x3C};
    byte b8[8]= {0x3C,0x00,0x00,0x00,0x00,0x18,0x3C,0x18};

    //Moving car
    printByte(b1);
    delay(50);
    printByte(b2);
    delay(50);
    printByte(b3);
    delay(50);
    printByte(b4);
    delay(50);
    printByte(b5);
    delay(50);
    printByte(b6);
    delay(50);
    printByte(b7);
    delay(50);
    printByte(b8);
    delay(50);

    //alternate pattern
    printByte(d1);
    delay(100);
```

3.10 LED Matrix 8X8 using MAX7219

```
printByte(d2);
    delay(100);

    //Arrow
    printByte(arrow_up);
    delay(2000);

    printByte(arrow_down);
    delay(2000);

    //Facial Expression
    printByte(smile);

    delay(1000);

    printByte(neutral);

    delay(1000);

    printByte(sad);

    delay(1000);

}

void printByte(byte character [])
{
  int i = 0;
  for(i=0;i<8;i++)
  {
    lc.setRow(0,i,character[i]);
  }
}
```

3.11 LED Matrix 8X8 using MAX7219 in Ping Pong Game

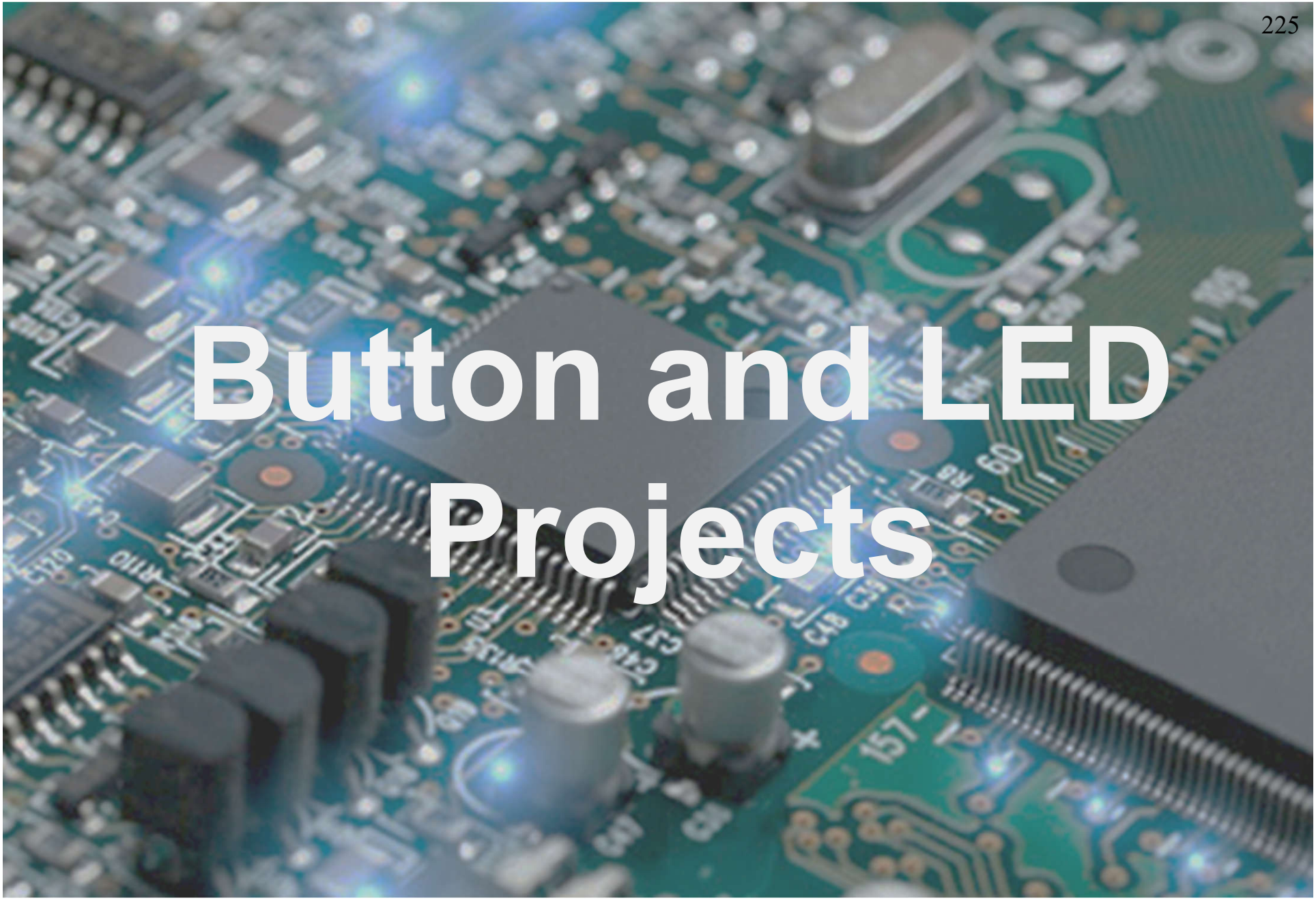


3.11 LED Matrix 8X8 using MAX7219 in Ping Pong Game

```
// Example 3.11 - LEDMatrixPongMax7219.ino
#include <LedControl.h>

LedControl myMatrix = LedControl(10, 8, 9, 1); // create an instance of a Matrix

int column = 1, row = random(8)+1; // decide where the ball will start
int directionX = 1, directionY = 1; // make sure it heads from left to right first
int paddle1 = 0, paddle1Val; // Pot pin and value
int speed = 300;
int counter = 0, mult = 10;

void setup()
{
        myMatrix.shutdown(0, false); // enable display
        myMatrix.setIntensity(0, 8); //  Set the brightness to medium
        myMatrix.clearDisplay(0); // clear the display
        randomSeed(analogRead(0));
}
```

3.11 LED Matrix 8X8 using MAX7219 in Ping Pong Game

```
void loop()
{
        paddle1Val = analogRead(paddle1);
        paddle1Val = map(paddle1Val, 200, 1024, 1,6);
        column += directionX;
           row += directionY;
        if (column == 6 && directionX == 1 && (paddle1Val == row || paddle1Val+1 ==
            row || paddle1Val+2 == row)) {directionX = -1;}
        if (column == 0 && directionX == -1 ) {directionX = 1;}
        if (row == 7 && directionY == 1 ) {directionY = -1;}
 if (row == 0 && directionY == -1 ) {directionY = 1;}
        if (column == 7) { oops();}
        myMatrix.clearDisplay(0); // clear the screen for next animation frame
        myMatrix.setLed(0, column, row, HIGH);
        myMatrix.setLed(0, 7, paddle1Val, HIGH);
        myMatrix.setLed(0, 7, paddle1Val+1, HIGH);
        myMatrix.setLed(0, 7, paddle1Val+2, HIGH);
        if (!(counter % mult)) {speed -= 5; mult * mult;}
        delay(speed);
        counter++;
}

void oops() {
        for (int x=0; x<3; x++) {
        myMatrix.clearDisplay(0);
        delay(250);
                for (int y=0; y<8; y++) {
                        myMatrix.setRow(0, y, 255);
                }
                delay(250);
        }
        counter=0; // reset all the values
        speed=300;
        column=1;
        row = random(8)+1; // choose a new starting location
}
```



Button and LED Projects

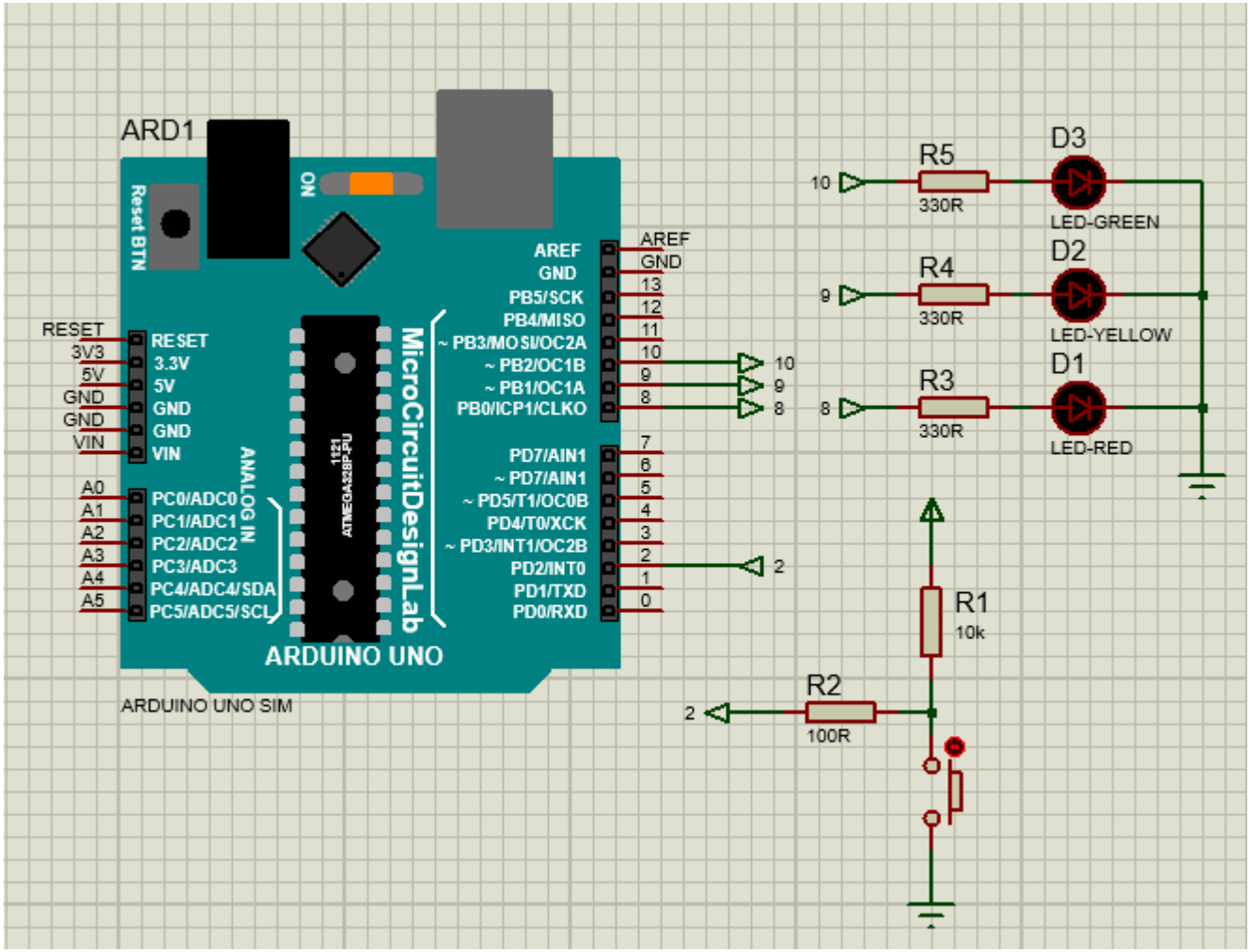
Button and LED Projects

Contents

4.0 Objectives

- In this tutorial, you will learn:
  - To understand how to implement LED_Button1.ino sketch.
  - To know how to make LED_Button2.ino sketch.
  - To become familiar using LED_Button3.ino sketch.
  - To understand how to make LED_Button4.ino sketch.
  - To know how to use LED_Button5.ino sketch.
  - To be able to start LED_Button6.ino sketch.
  - To appreciate how use LED_Button7.ino sketch.
  - To appreciate how use LED_Button8.ino sketch.

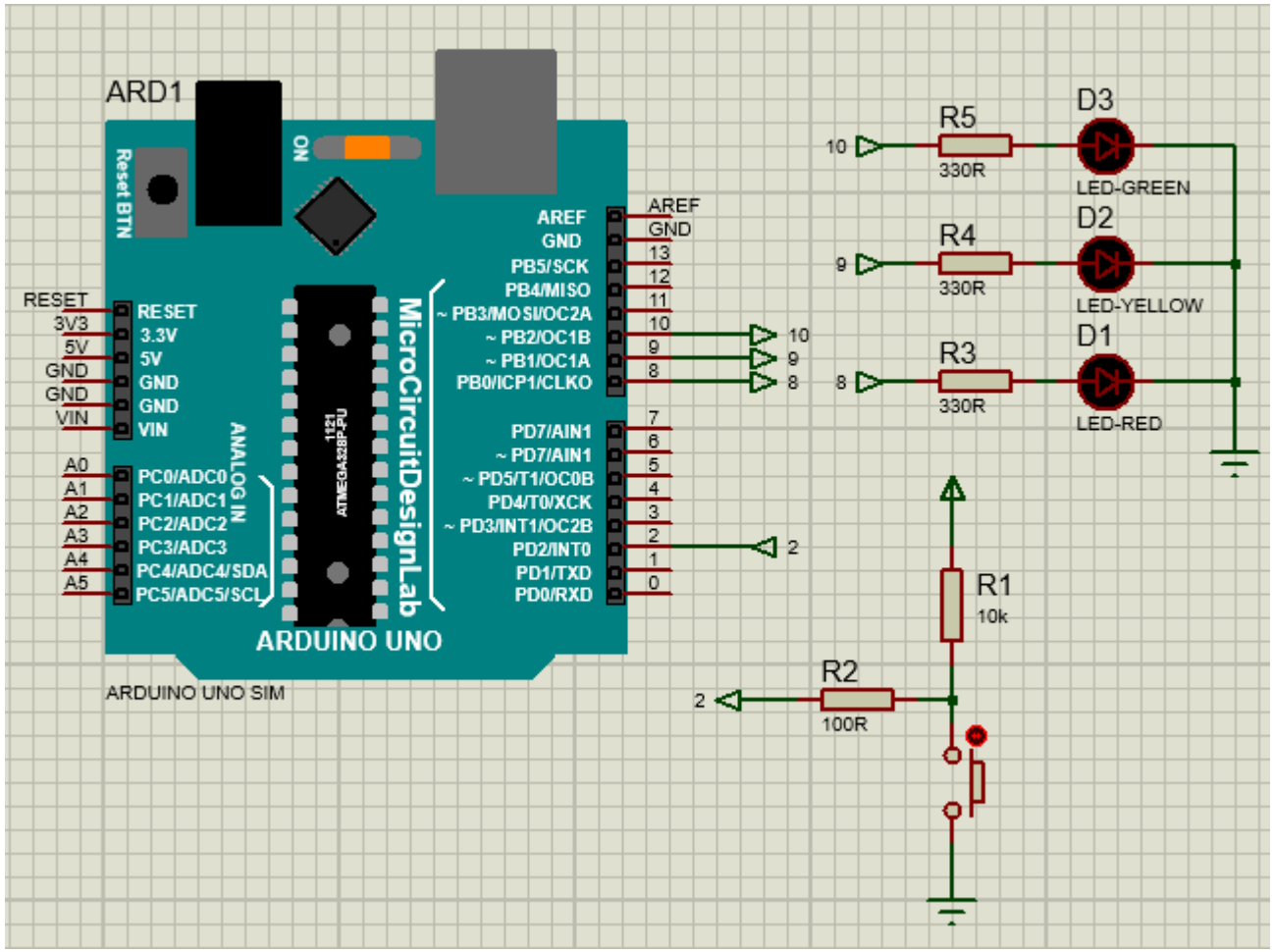
4.1 LED Button 1



4.1 LED Button 1

```
//  Example 4.1 LED_Button1.ino
// Arduino Pins to be used
int pin_LED8 = 8;                       // LED in pin 10
int pin_Button2 = 2;                    // Push Button in pin 2

void setup()
{

    pinMode(pin_LED8, OUTPUT);          // Initialize pin 10 to output
    digitalWrite(pin_LED8,LOW);         // Initialize pin 10 LOW during start
    pinMode(pin_Button2, INPUT);        // Initialize pin 2 as digital input
}

void loop()
{
    if ( digitalRead(pin_Button2) == LOW)// If push button was pressed
    {
       digitalWrite(pin_LED8, HIGH);     // LED pin 8 ON
    }
    else
    {
       digitalWrite(pin_LED8, LOW);      // If not pressed, pin 10 OFF
    }
}
```

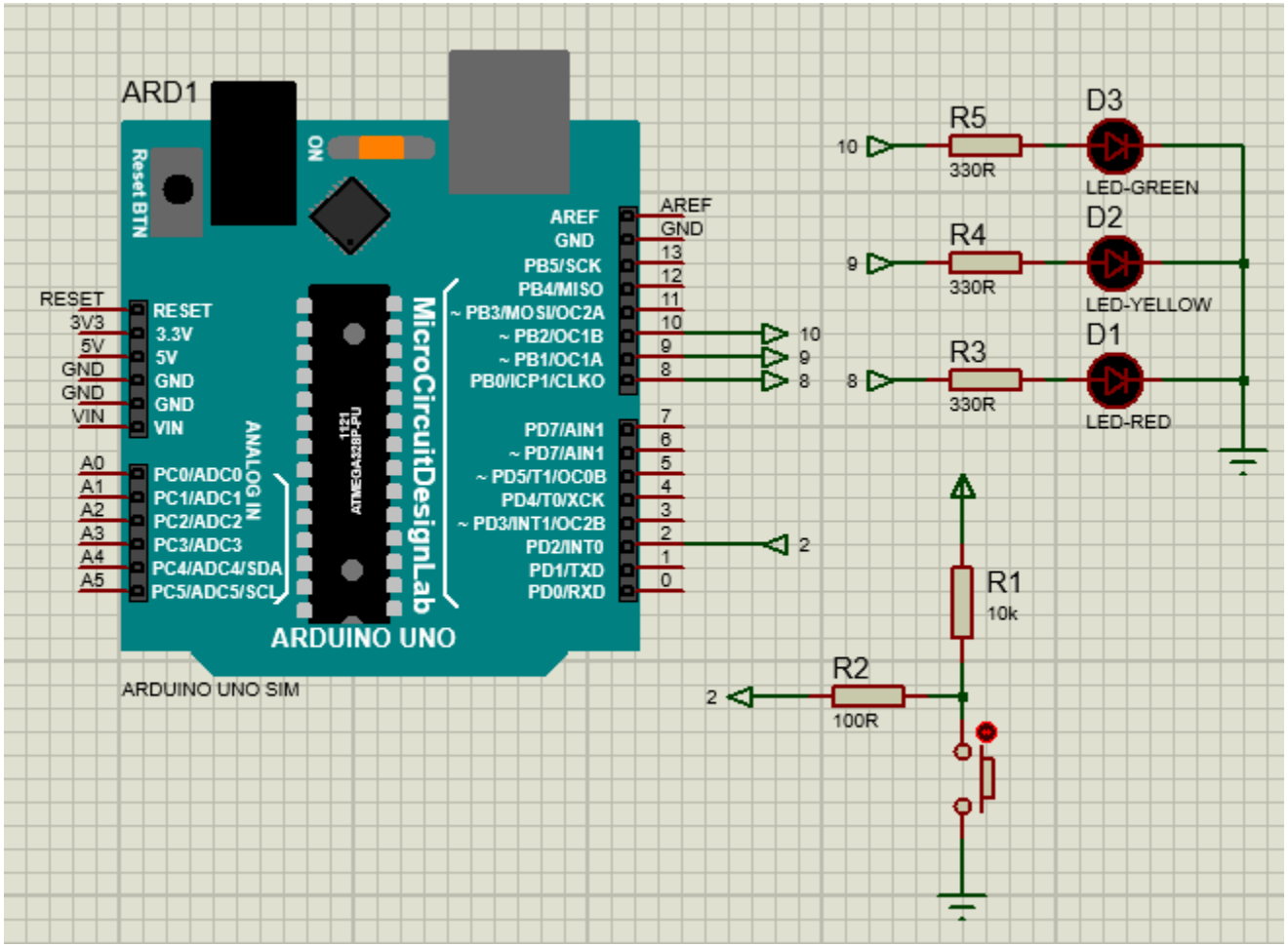
4.2 LED Button 2



4.2 LED Button 2

```
//  Example 4.2 - LED_Button2.ino
//  Turning an LED in pin 8 on and off using a button switch in pin 2

// Arduino pins being used
int pin_LED8 = 8;
int pin_Button2 = 2;

// Variables new and old button states
boolean oldButton8State = HIGH;
boolean newButton8State = HIGH;

void setup()
{
    pinMode(pin_LED8, OUTPUT);
    digitalWrite(pin_LED8,LOW);
    pinMode(pin_Button2, INPUT);
}

void loop()
{
    newButton8State = digitalRead(pin_Button2);

    if ( newButton8State != oldButton8State )
    {
       if ( newButton8State == LOW )
       {
        digitalWrite(pin_LED8, HIGH);
       }
       else
       {
        digitalWrite(pin_LED8, LOW);
       }

       oldButton8State = newButton8State;
    }
}
```

4.3 LED Button 3



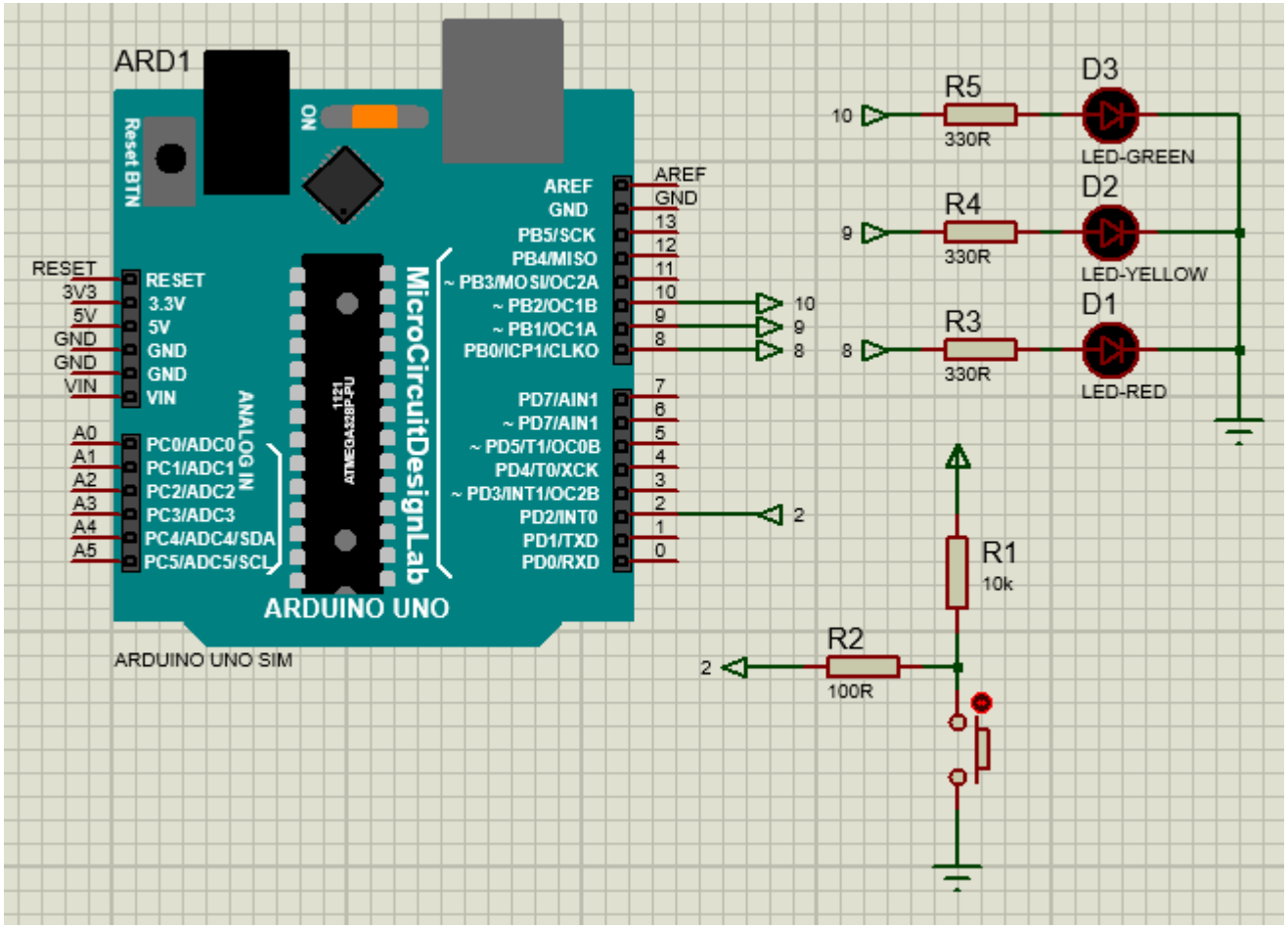
4.3 LED Button 3

```
//  Example 4.3 - LED_Button3.ino
//  Button switch as a toggle switch to turn an LED on or off


// Define the pins being used
int pin_LED8 = 8;
int pin_Button8 = 2;

// Variables of new and old button states
boolean oldButton8State = HIGH;
boolean newButton8State = HIGH;

boolean LED8state = LOW;

void setup()
{
    pinMode(pin_LED8, OUTPUT);
    digitalWrite(pin_LED8,LOW);
    pinMode(pin_Button8, INPUT);
}
```

4.3 LED Button 3

```
void loop()
{
    newButton8State = digitalRead(pin_Button8);

    if ( newButton8State != oldButton8State )
    {
       // if pin_Button8 pressed
       if ( newButton8State == LOW )
       {
           if ( LED8state == LOW )
           { digitalWrite(pin_LED8, HIGH);
           LED8state = HIGH;
           }
           else
           { digitalWrite(pin_LED8, LOW);
           LED8state = LOW;
           }
       }
       oldButton8State = newButton8State;
    }
}

}
```

4.4 LED Button 4



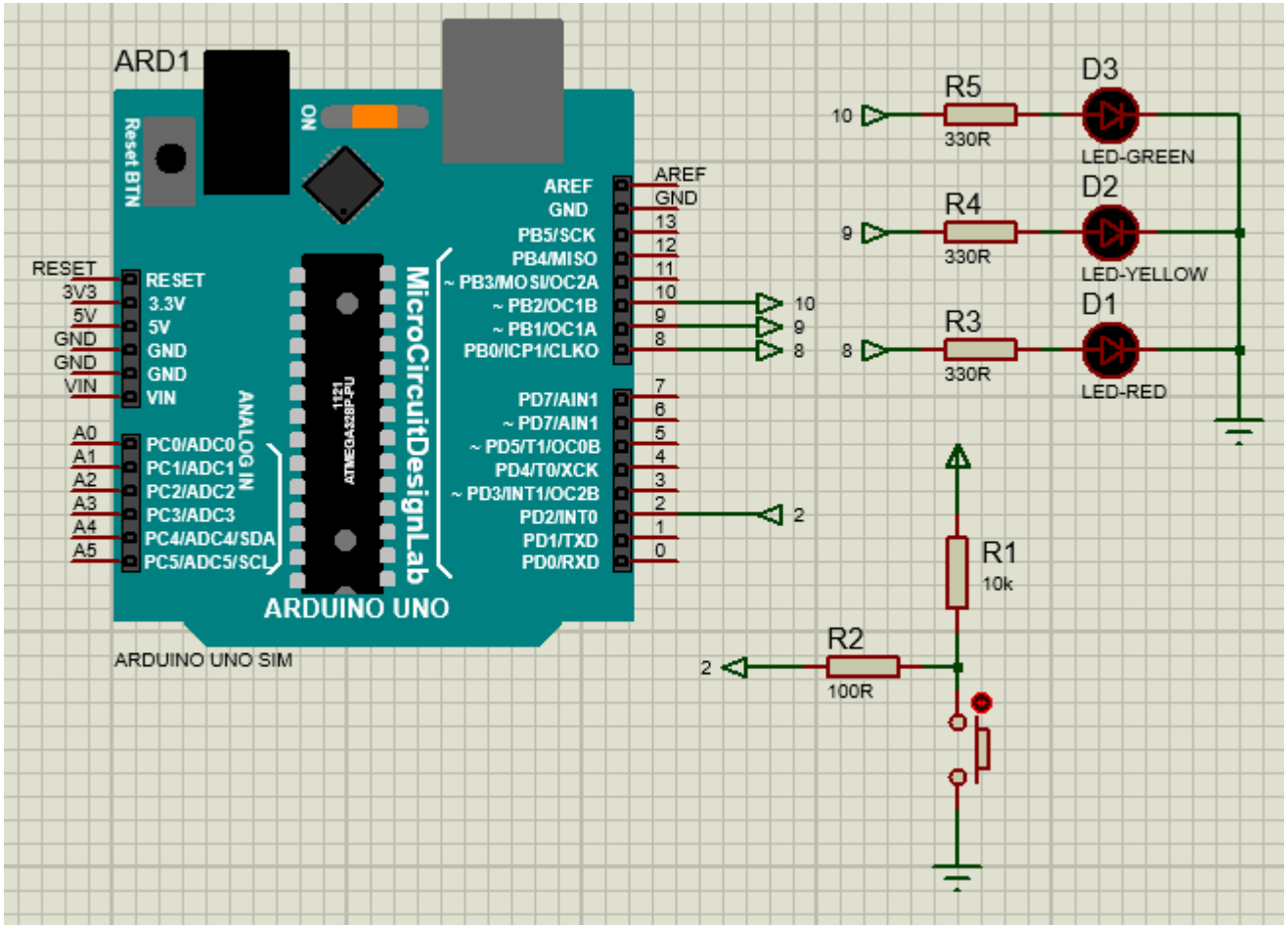
4.4 LED Button 4

```
//  Example 4.4 - Led_Button4.ino
//  Using a button switch as a toggle switch to turn an LED on or off
//  with a simple debounce

// Define the pins being used
int pin_LED8 = 8;
int pin_Button2 = 2;

// new and old switch states
boolean oldButton2State = HIGH;
boolean newButton2State1 = HIGH;
boolean newButton2State2 = HIGH;
boolean newButton2State3 = HIGH;

// LED8 state
boolean LED8state = LOW;

void setup()
{
    pinMode(pin_LED8, OUTPUT);
    digitalWrite(pin_LED8,LOW);
    pinMode(pin_Button2, INPUT);
}
```

4.4 LED Button 4

```
void loop()
{
    newButton2State1 = digitalRead(pin_Button2);
    delay(1);
    newButton2State2 = digitalRead(pin_Button2);
    delay(1);
    newButton2State3 = digitalRead(pin_Button2);

    // if all 3 values are the same we can continue
    if (  (newButton2State1==newButton2State2) && (newButton2State1==newButton2State3) )
    {

        if ( newButton2State1 != oldButton2State )
        {

           // has the button switch been closed?
           if ( newButton2State1 == LOW )
           {
               if ( LED8state == LOW )
               {
                  digitalWrite(pin_LED8, HIGH);
                  LED8state = HIGH;
               }
               else
               {
                  digitalWrite(pin_LED8, LOW);
                  LED8state = LOW;
               }
           }
           oldButton2State = newButton2State1;
        }
    }
}
```

4.5 LED Button 5



4.5 LED Button 5

```
//  Example 4.5 - LED_Button5.ino
//
//  Example of using a single button switch to set multiple states or conditions
//
//  D10-green LED
//  D9-yellow LED
//  D8-red LED
//  D2-push button
//
//  state holds the current status.
//  0 = all off.
//  1 = green LED
//  2 = yellow LED
//  3 = red LED

// Define the pins being used
int pin_LED10green = 10;
int pin_LED9yellow = 9;
int pin_LED8red = 8;

int pin_Button2 = 2;

// new and old switch states
boolean oldButtonState = HIGH;
boolean newButtonState1 = HIGH;
boolean newButtonState2 = HIGH;
boolean newButtonState3 = HIGH;

byte state = 0;
```

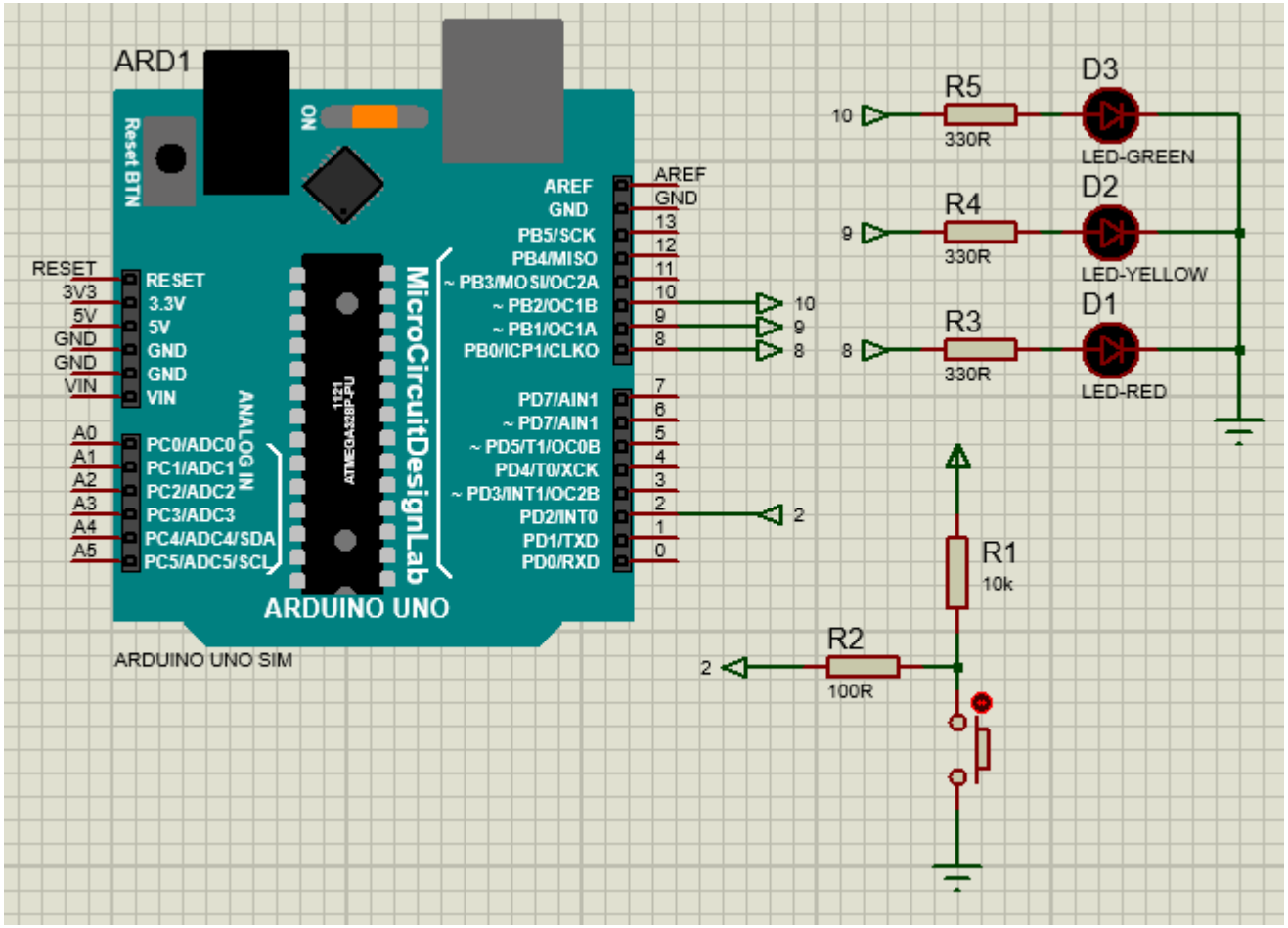
4.5 LED Button 5

```
void setup()
{

    pinMode(pin_LED10green, OUTPUT);    digitalWrite(pin_LED10green,LOW);
    pinMode(pin_LED9yellow, OUTPUT);   digitalWrite(pin_LED9yellow,LOW);
    pinMode(pin_LED8red, OUTPUT);      digitalWrite(pin_LED8red,LOW);
    pinMode(pin_Button2, INPUT);
}

void loop()
{
    newButtonState1 = digitalRead(pin_Button2);
    delay(1);
    newButtonState2 = digitalRead(pin_Button2);
    delay(1);
    newButtonState3 = digitalRead(pin_Button2);

    // if all 3 values are the same we can continue
    if (  (newButtonState1==newButtonState2) && (newButtonState1==newButtonState3) )
    {

        if ( newButtonState1 != oldButtonState )
        {

           // has the button switch been closed?
           if ( newButtonState1 == LOW )
           {
                // increase the value of state
                state++;
                if (state > 3) { state = 0; }
```


4.5 LED Button 5

```
// turn all LEDs off. Doing it this way means we do not need to care about the individual LEDs
                // simply turn them all off and then turn on the correct one.
                digitalWrite(pin_LED10green, LOW);
                digitalWrite(pin_LED9yellow, LOW);
                digitalWrite(pin_LED8red, LOW);

                // Turn on the next LED
                // Because the value of state does not change while we are testing it we don't
need to use else if
                if (state==1) { digitalWrite(pin_LED10green, HIGH);  }
                if (state==2) { digitalWrite(pin_LED9yellow, HIGH);  }
                if (state==3) { digitalWrite(pin_LED8red, HIGH);  }

            }
            oldButtonState = newButtonState1;
        }
    }
}
```

4.6 LED Button 6



4.6 LED Button 6

```
//  Example 4.6 - LED_Button6.ino
//  Example of using a single button switch to set multiple states or conditions
//
//  Pins
//  D10-green LED
//  D9-yellow LED
//  D8-red LED
//  D2-push button
//
//  state holds the current status.
//  0 = all off.
//  1 = green LED
//  2 = yellow LED
//  3 = red LED

// Define the pins being used fro the LEDs green/yellow/red
char LED_Pins_array[] = { 10, 9, 8};

// Array to hold the LED sequence. green, yellow, red, yellow, green.
// position 0 is not used (considered not good practice but keeps the code easy to understand)
char LED_Sequence_array[] = { 10, 9, 8, 9};
byte squenceLength = 4;

int pin_Button2 = 2;
```

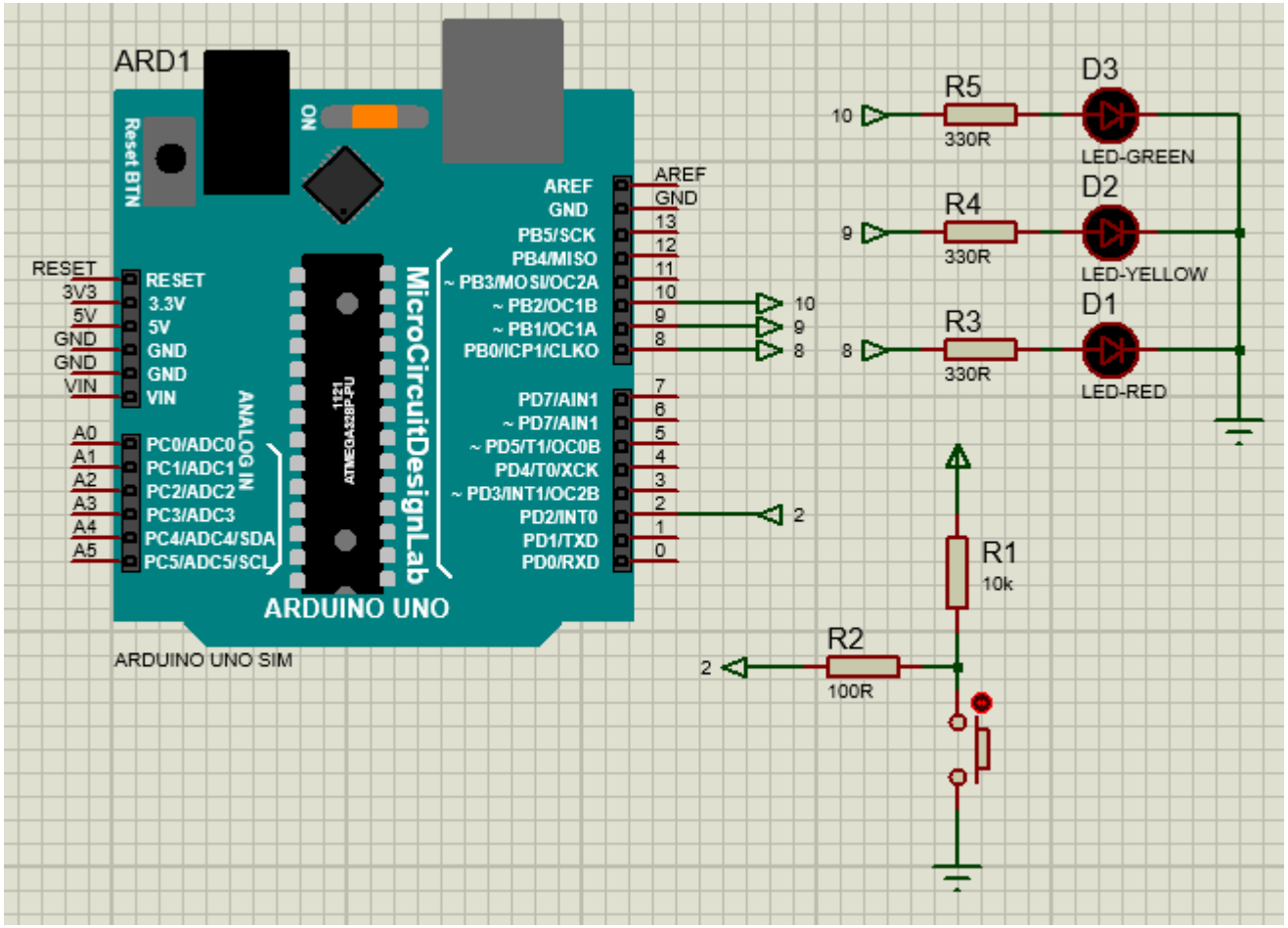
4.6 LED Button 6

```
// variables to hold the new and old switch states
boolean oldSwitchState = HIGH;
boolean newSwitchState1 = HIGH;
boolean newSwitchState2 = HIGH;
boolean newSwitchState3 = HIGH;

byte state = -1;

void setup()
{

    for (byte i=0; i< 3; i++)
    {
         pinMode(LED_Pins_array[i], OUTPUT);
         digitalWrite(LED_Pins_array[i],LOW);
    }
         pinMode(pin_Button2, INPUT);
}
```

4.6 LED Button 6

```
void loop()
{
    newSwitchState1 = digitalRead(pin_Button2);
    delay(1);
    newSwitchState2 = digitalRead(pin_Button2);
    delay(1);
    newSwitchState3 = digitalRead(pin_Button2);

    // if all 3 values are the same we can continue
    if (  (newSwitchState1==newSwitchState2) && (newSwitchState1==newSwitchState3) )
    {
        if ( newSwitchState1 != oldSwitchState )
        {

           // has the button switch been closed?
           if ( newSwitchState1 == LOW )
           {
                state++;
                if (state > (squenceLength -1) ) { state = 0; }

                // turn all LEDs off. Doing it this way means we do not need to care about the
individual LEDs
                // simply turn them all off and then turn on the correct one.
                for (byte i=0; i< 3; i++)
                {
                     digitalWrite(LED_Pins_array[i],LOW);
                }

                // Turn on the next LED
                digitalWrite(LED_Sequence_array[state],HIGH);
           }
           oldSwitchState = newSwitchState1;
        }
    }
}
```

4.7 LED Button 7



4.7 LED Button 7

```
//  Example 4.7 - LED_Button7.ino
//
//  Example of using a button switch as a toggle switch to turn a blinking LED on or off
//
//  Pins
//  D8 - LED
//  D2 - Push Button
//

// Define the pins being used
int pin_LED8 = 8;
int pin_Button2 = 2;


// variables to hold the new and old switch states
boolean oldSwitchState = HIGH;
boolean newSwitchState1 = HIGH;
boolean newSwitchState2 = HIGH;
boolean newSwitchState3 = HIGH;

// variables to hold the times
unsigned long timeNow = 0;
unsigned long timePrev = 0;
unsigned int timeWait = 100;

// variables used to control the LED
boolean flashingLEDisON = false;
boolean LEDstatus = LOW;
boolean keyPressed = false;
```

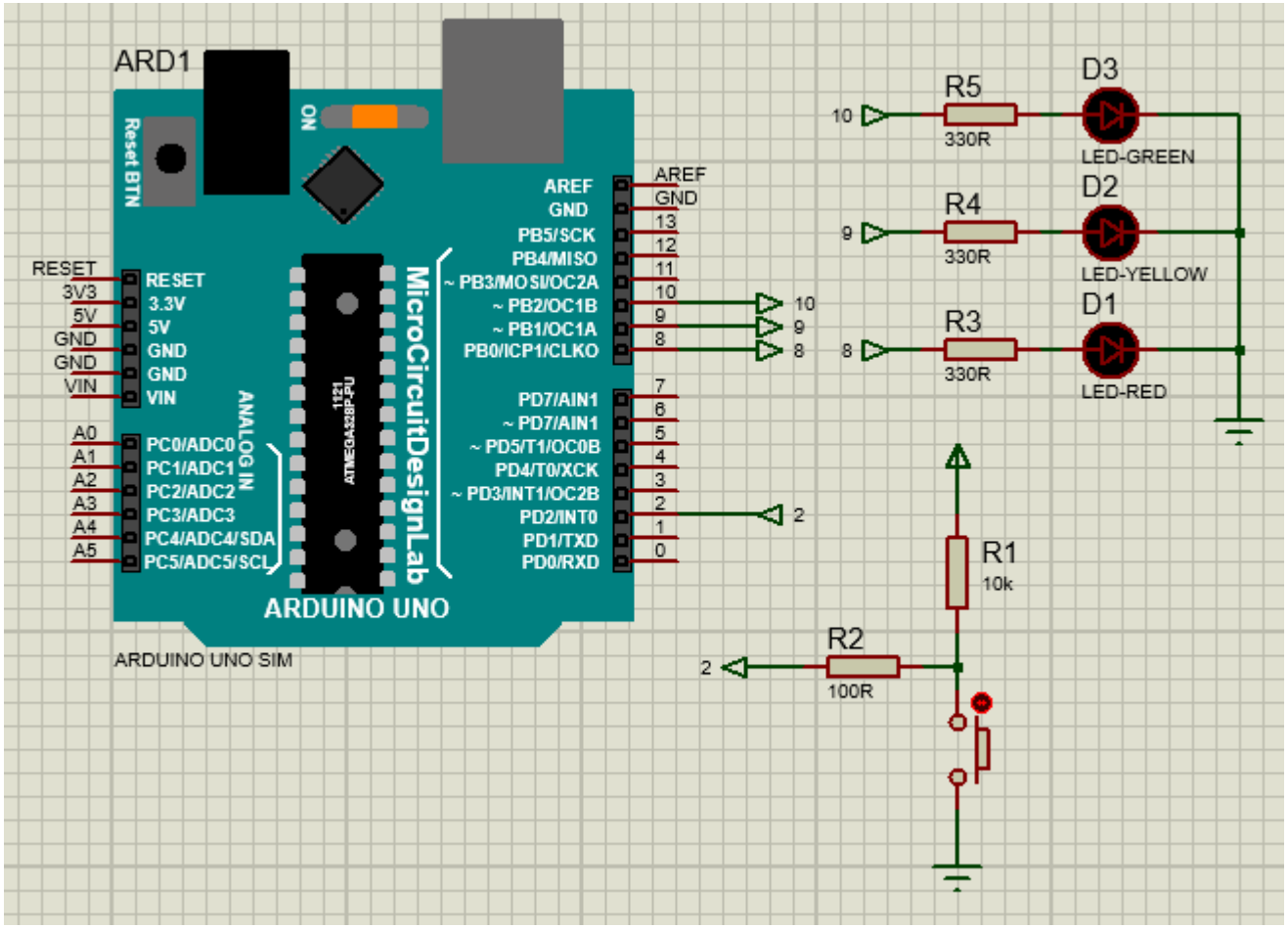
4.7 LED Button 7

```
void setup()
{


    pinMode(pin_LED8, OUTPUT);
    digitalWrite(pin_LED8,LOW);
    pinMode(pin_Button2, INPUT);
}

void loop()
{
    newSwitchState1 = digitalRead(pin_Button2);      delay(1);
    newSwitchState2 = digitalRead(pin_Button2);      delay(1);
    newSwitchState3 = digitalRead(pin_Button2);

    if (  (newSwitchState1==newSwitchState2) && (newSwitchState1==newSwitchState3) )
    {
        if ( newSwitchState1 != oldSwitchState )
        {
            if ( newSwitchState1 == LOW ) { keyPressed = true; } else { keyPressed =
false; }
            oldSwitchState = newSwitchState1;
        }
    }
```

4.7 LED Button 7

```
if ( keyPressed )
    {
         // turn on or turn off the blinking LED
         if ( flashingLEDisON == false)
         {
             flashingLEDisON = true;
         }
         else
         {
              flashingLEDisON = false;
              // the LED may be on so to be safe we turn it off. If you wished you could check
LEDstatus
              LEDstatus = LOW;  digitalWrite(pin_LED8, LEDstatus);
         }
         keyPressed = false;
    }


    // if the blinking LED is on. See if it is time to blink it
    if ( flashingLEDisON == true )
    {
        timeNow = millis();
        if (timeNow-timePrev >= timeWait )
        {
              timePrev = timeNow;
              if (LEDstatus == LOW) { LEDstatus = HIGH; } else { LEDstatus = LOW; }
              digitalWrite(pin_LED8, LEDstatus);
        }
    }
}
```

4.8 LED Button 8



4.8 Button LED 8

```
//  Example 4.8 - LED_Button8.ino
//
//  Example of using a button switch as a toggle switch to turn a blinking LED on or off
//  now using functions
//
//  Pins
//  D8-LED
//  D2-push button
//

// Define the pins being used
int pin_LED8 = 8;
int pin_Button2 = 2;


// variables to hold the new and old switch states
boolean oldSwitchState = HIGH;

// variables to hold the times
unsigned long timeNow = 0;
unsigned long timePrev = 0;
unsigned int timeWait = 100;

// variables used to control the LED
boolean flashingLEDisON = false;
boolean LEDstatus = LOW;
boolean keyPressed = false;
```


4.8 Button LED 8

```
void setup()
{


    pinMode(pin_LED8, OUTPUT);
    digitalWrite(pin_LED8,LOW);

    pinMode(pin_Button2, INPUT);
}

void loop()
{
    keyPressed = checkButtonSwitch();
    if ( keyPressed )
    {
        keyPressed = false;
        startAndStop();
    }
    if ( flashingLEDisON == true )  { blinkTheLED();   }
}
```

4.8 Button LED 8

```
boolean checkButtonSwitch()
{
    boolean key = false;

    boolean newSwitchState1 = digitalRead(pin_Button2);      delay(1);
    boolean newSwitchState2 = digitalRead(pin_Button2);      delay(1);
    boolean newSwitchState3 = digitalRead(pin_Button2);

    if (  (newSwitchState1==newSwitchState2) && (newSwitchState1==newSwitchState3) )
    {
        if ( newSwitchState1 != oldSwitchState )
        {
            if ( newSwitchState1 == LOW ) { key = true; } else { key =  false; }
            oldSwitchState = newSwitchState1;
        }
    }
    return key;
}
```

4.8 Button LED 8

```
void startAndStop( )
{
     // turn on or turn off the blinking LED
     if ( flashingLEDisON == false)
     {
         flashingLEDisON = true;
     }
     else
     {
         flashingLEDisON = false;
         // the LED may be on so we turn it off just is case
         LEDstatus = LOW;
         digitalWrite(pin_LED8, LEDstatus);
     }
}


void blinkTheLED()
{
    timeNow = millis();
    if (timeNow-timePrev >= timeWait )
    {
          timePrev = timeNow;
          if (LEDstatus == LOW) { LEDstatus = HIGH; } else { LEDstatus = LOW; }
          digitalWrite(pin_LED8, LEDstatus);
    }
}
```

Thank You Next is Display Technologies!!!