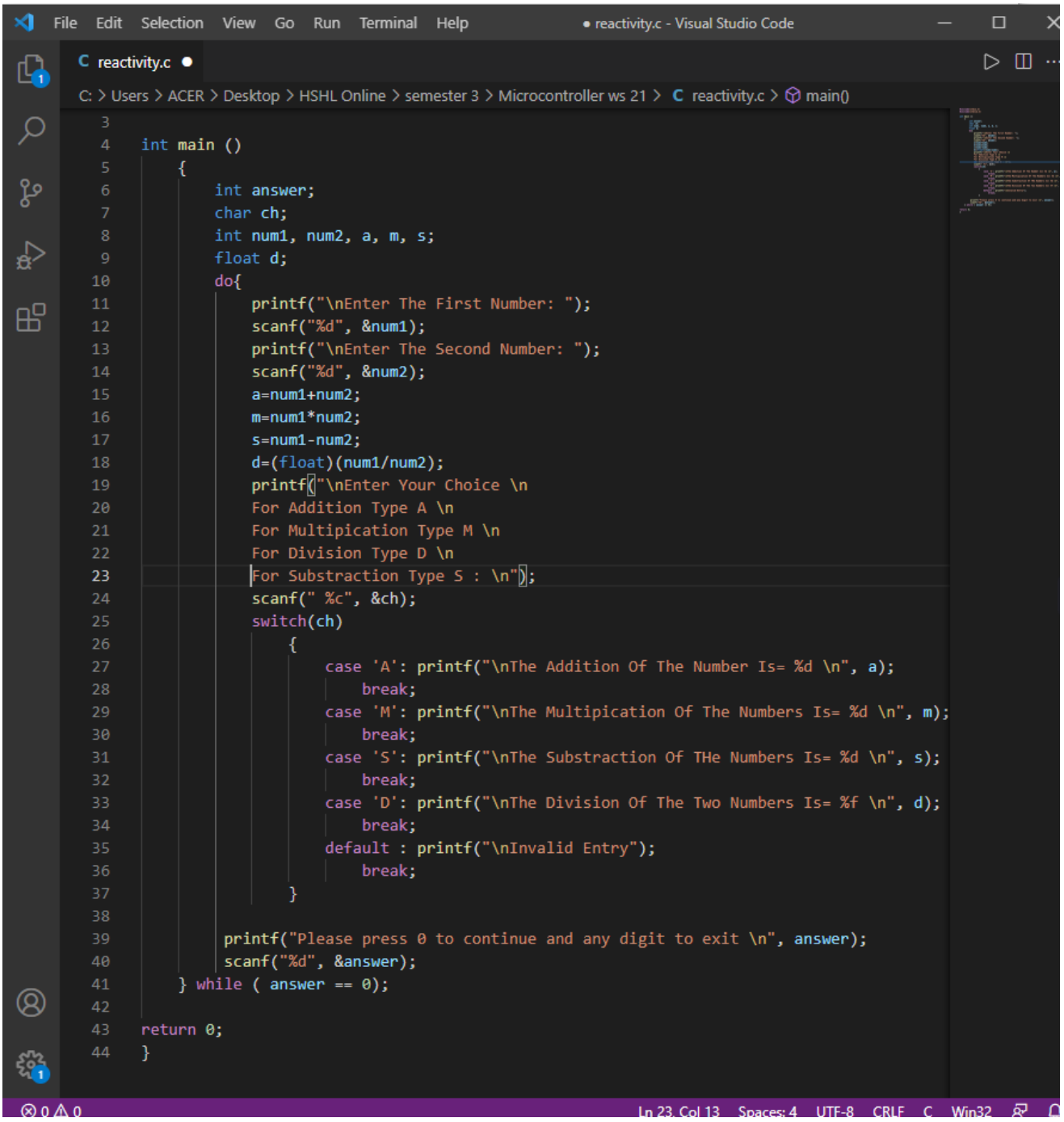
Microcontroller Assignments

01:

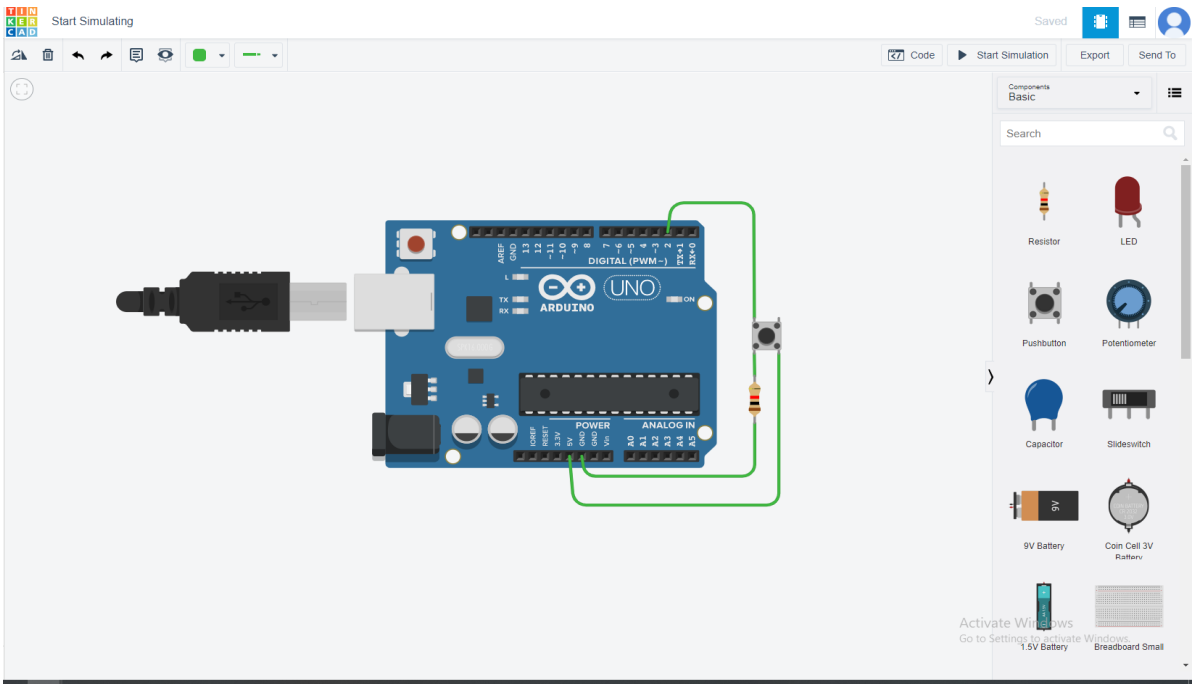
1. Reactive Systems' Implementation:

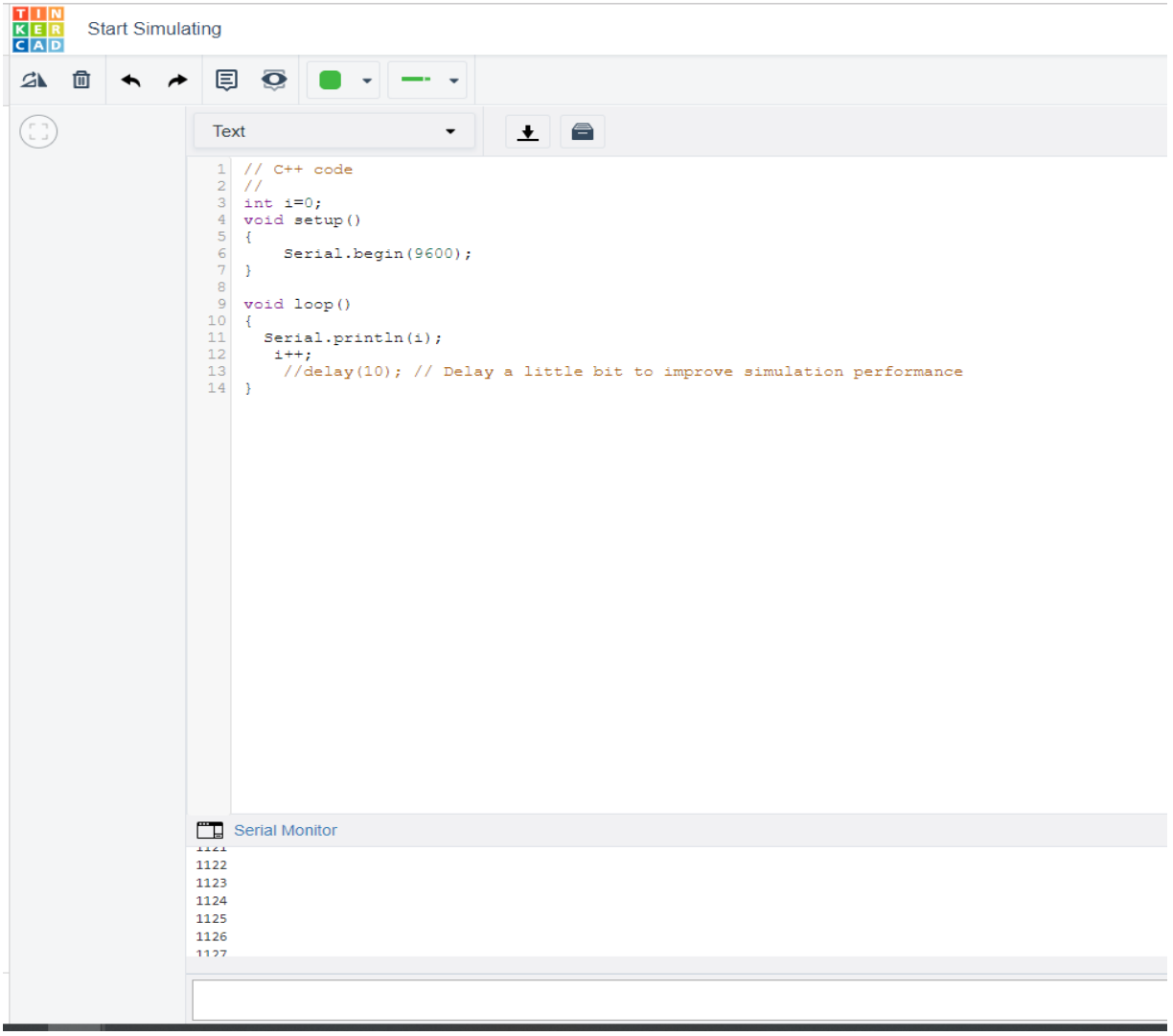
A. As a C++ Program:



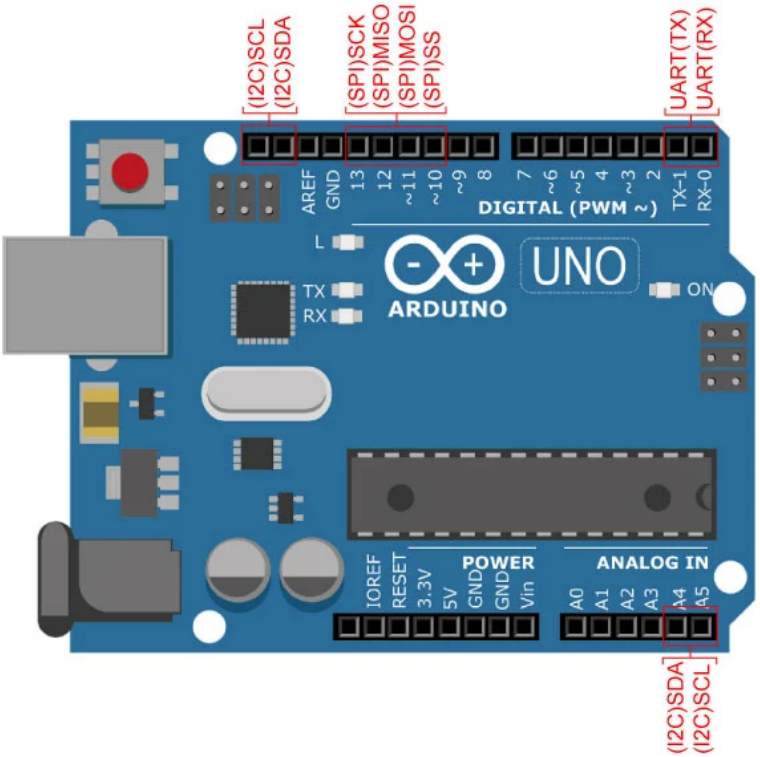

```c
int main ()
    {
        int answer;
        char ch;
        int num1, num2, a, m, s;
        float d;
        do{
            printf("\nEnter The First Number: ");
            scanf("%d", &num1);
            printf("\nEnter The Second Number: ");
            scanf("%d", &num2);
            a=num1+num2;
            m=num1*num2;
            s=num1-num2;
            d=(float)(num1/num2);
            printf("\nEnter Your Choice \n
            For Addition Type A \n
            For Multipication Type M \n
            For Division Type D \n
            For Substraction Type S : \n");
            scanf(" %c", &ch);
            switch(ch)
                {
                    case 'A': printf("\nThe Addition Of The Number Is= %d \n", a);
                        break;
                    case 'M': printf("\nThe Multipication Of The Numbers Is= %d \n", m);
                        break;
                    case 'S': printf("\nThe Substraction Of THe Numbers Is= %d \n", s);
                        break;
                    case 'D': printf("\nThe Division Of The Two Numbers Is= %f \n", d);
                        break;
                    default : printf("\nInvalid Entry");
                        break;
                }

           printf("Please press 0 to continue and any digit to exit \n", answer);
           scanf("%d", &answer);
    } while ( answer == 0);

return 0;
}
```

B. As an Arduino Program:





- **Communication protocols:**
  There are several communication protocols for arduino and each one has different specifications:
    - Universal Asynchronous Reception and Transmission(UART): It allows arduino to communicate with a serial device through a digital pin 0 and a digital pin 1 with another computer through USB port.
    - Inter-integrated-circuit (I2C): It is a communication serial protocol dedicated to microcontrollers.
    - Serial Peripheral Interface (SPI): It's a serial communication protocol that allows microcontrollers to communicate together.



- **Interrupts:**
  An interrupt is a signal transmitted to the microprocessor to stop immediately an actual activity and execute a high priority task.

Interrupts could be:

- ❖ Timer interrupt
- ❖ External interrupt
- ❖ Pin-change interrupt

2. Deadlines

- A deadline is the time at which a specific task should be completed. There is two types of deadlines:
  - ❖ **Hard deadline:** in case of missing it could be catastrophic**.**
  - ❖ **Soft deadline:** if missed, the still will be functional but there will be a degradation in its quality.

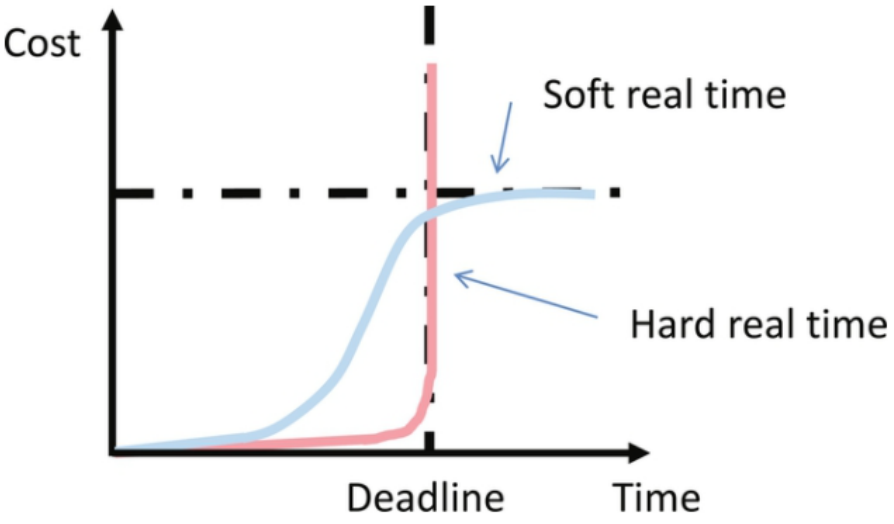


FIG. 10 A comparison between hard and soft real-time systems.

- **Tools and techniques that allows us to meet deadlines:**
  - ❖ **Continuous Testing:**

  To make sure that the system is continuously behaving as it is required regardless of the code changes.
  - ❖ **Interrupts:** In order to execute an immediate function with high priority within a specific time.

3. Embedded Systems and HW platforms:

"An embedded system is a specialized computer system that is usually integrated as part of a larger system. An embedded system consists of a combination of hardware

and software components to form a computational engine that will perform a specific function.”

An embedded system performs in an environment that needs to be reactive as well as time-limited. An Embedded system is a combination of hardware (that provides the required performance of the system) and software (which contains the different features that will ensure a flexible use of the system).

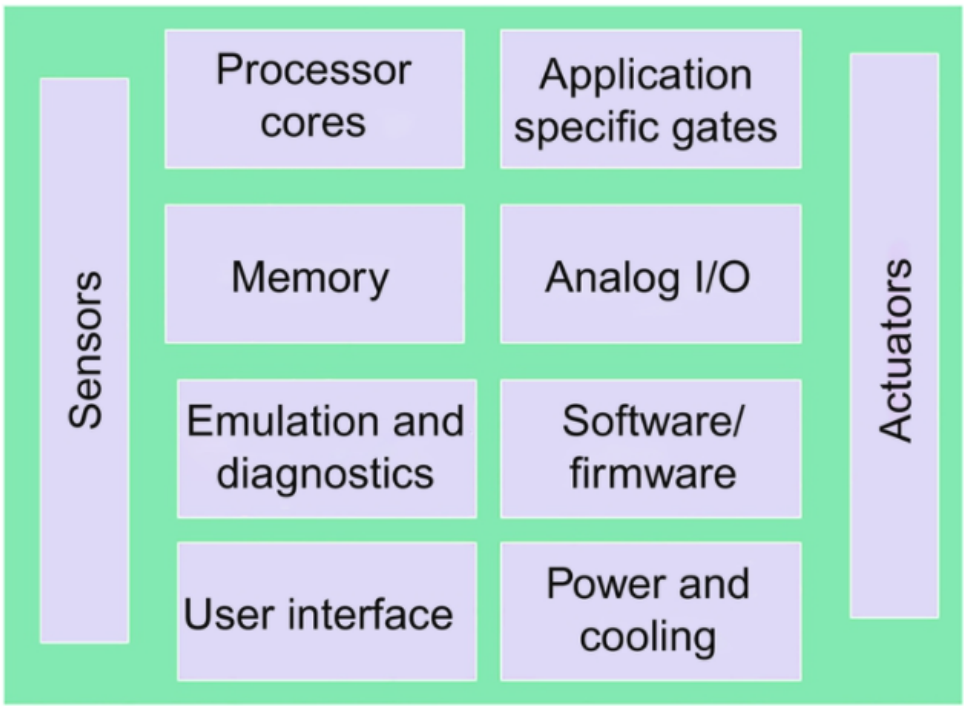
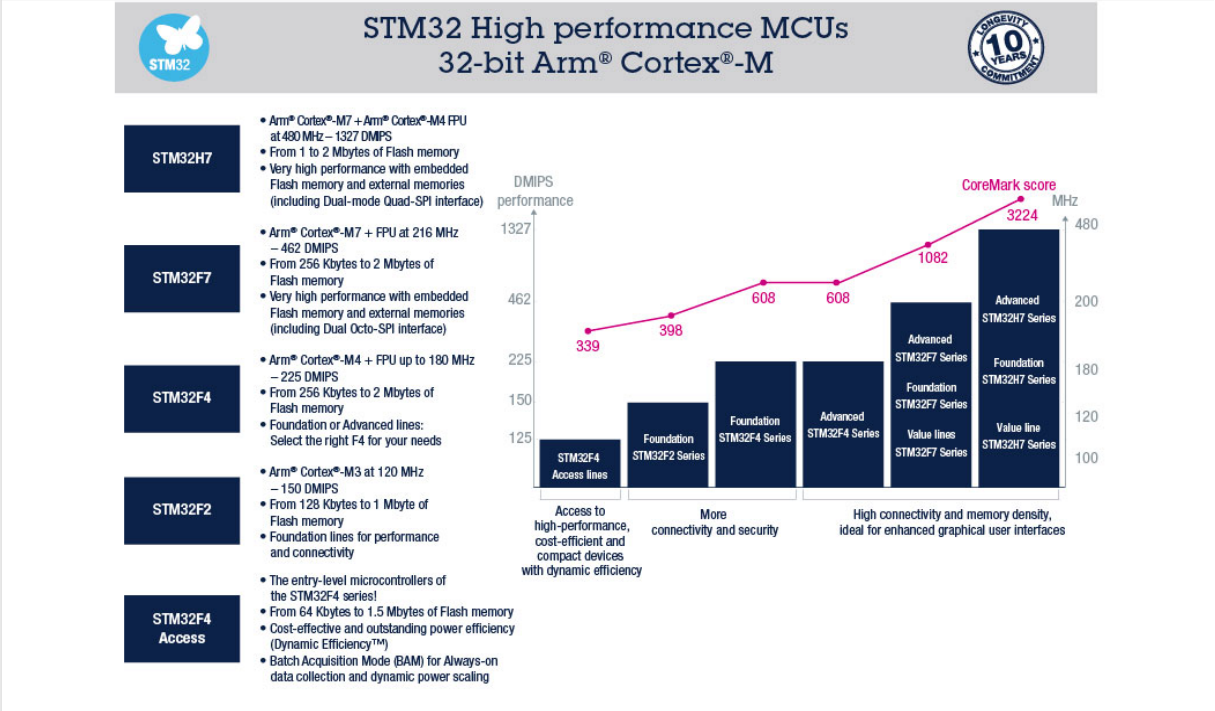


FIG. 5 Components of a typical embedded system.

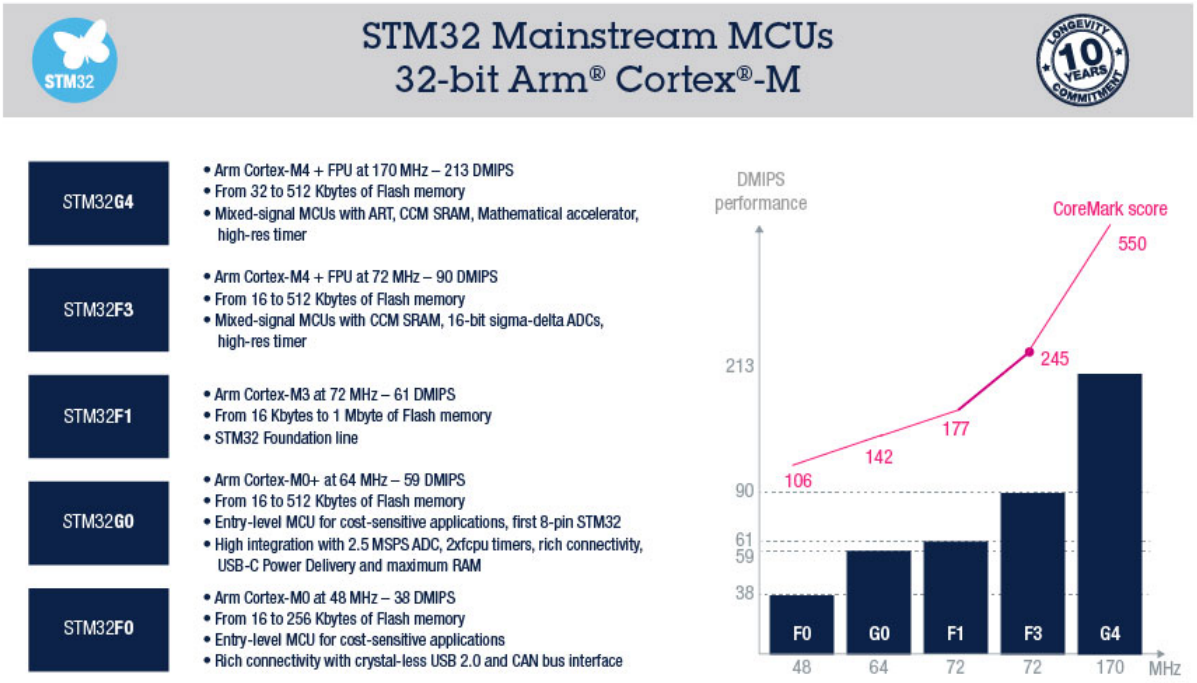
- **The different characteristics of an embedded system:**
  - Monitoring and reacting to the environment
  - Control the environment.
  - Processing information
  - Application specific.
  - Optimized for the application
  - Resource constrained
  - Real time.
  - Multirate
- **Hw Platforms:**
  - **STM32 Family:**
    - ➢ **STM32 High Performance MCUs:**
      - STM32F Series
      - STM32F4 series

- STM32F2 series



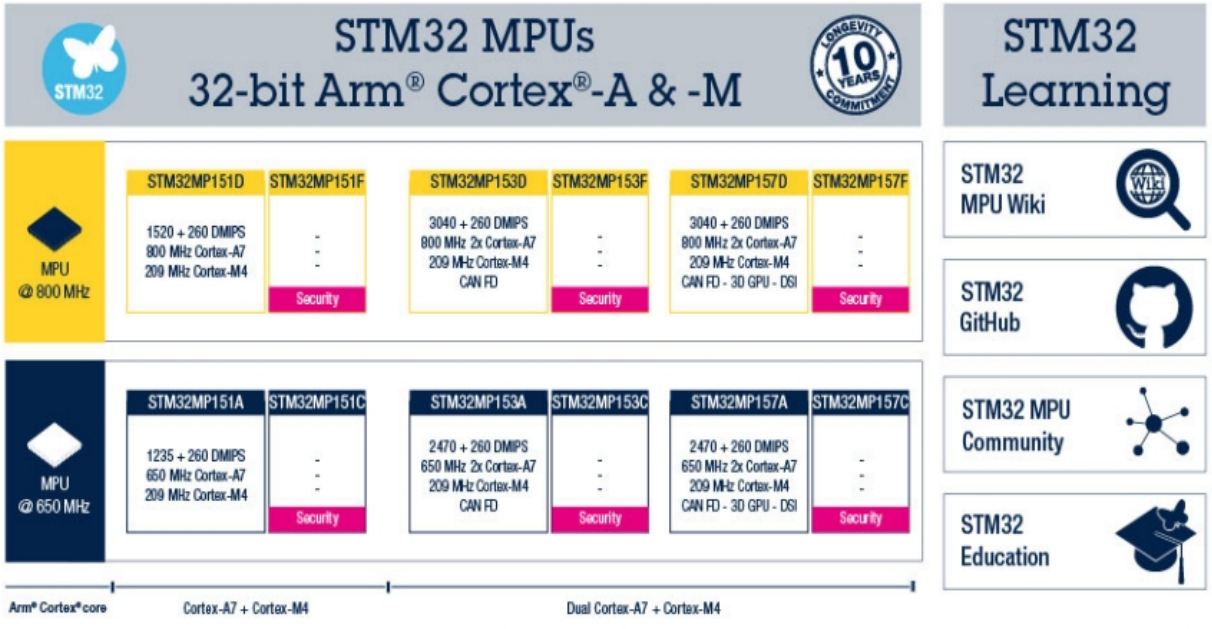
➢ **STM32 Mainstream MCUs:**

- STM32F0 Series
- STM32G0 Series
- STM32F1 Series
- STM32F3 Series
- STM32G4 Series



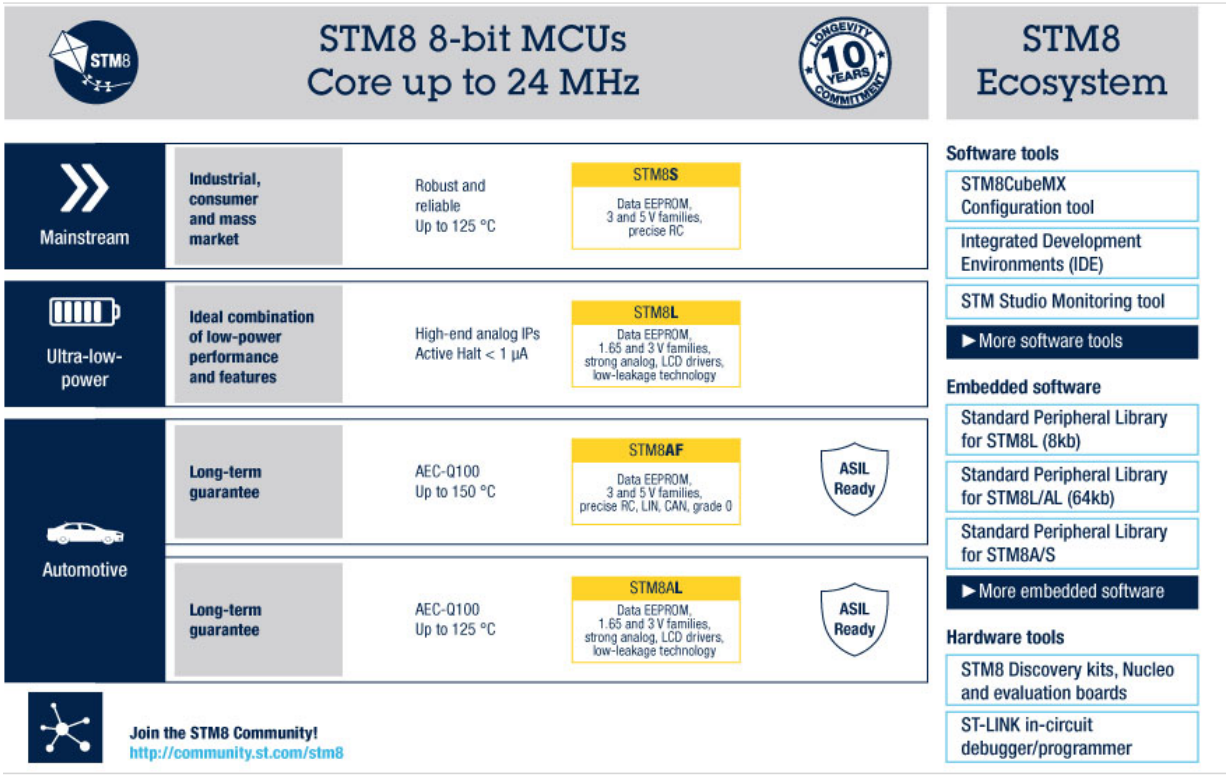
➢ **STM32 Arm Cortex MPUs:**

- STM32MP1 Series



➢ **STM8 8-bit MCUs:**

- STM8S, mainstream MCUs
- STM8L, ultra-low-power MCUs
- STM8AF and STM8AL, automotive MCUs



4. Continuous real world example to a discrete example : ?

02:

1. Fulfilling Embedded Systems Characteristics:

- **Reactive systems**

  Continuously interacting with the environment ,Non terminating, Stimulus/response

  interaction (must be able to respond to interrupts)

- **Real-time systems**

  Means the response of a Real-Time system must take place within a specific time frame.

- **Continuous/discrete/hybrid systems**

  A discrete system ,in which the state variable changes only at a discrete set of points in time.

  A continuous system ,in which the state variable changes continuously over time.

Hybrid: Continuous and discrete elements are valid.

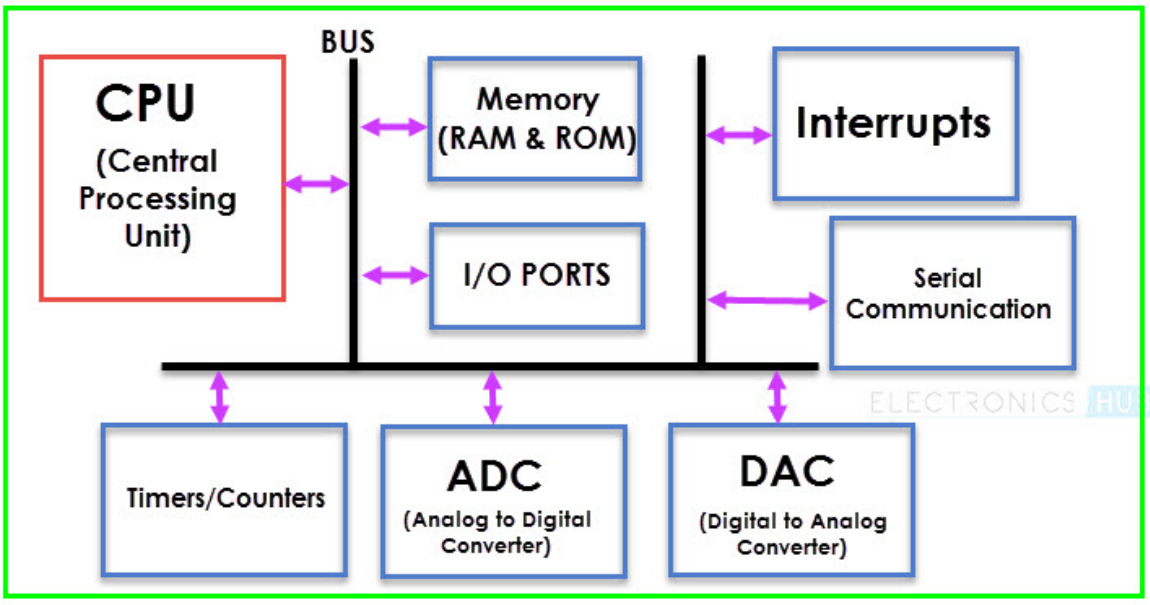
- **Embedded systems**
  It is a software embedded into technical system interact with physical components (Actuators,
  sensors) to control specific hardware.
- **Dependable systems**
  Can rely on the system in two aspects, that it performs according to its service specification and
  that the system avoids hazards.
- **Distributed systems**
  It contains multiple nodes that are physically separate but linked together using the network

2. Attributes of dependability:

- **reliability**
  Reliability is the probability that the system will conform to its specification throughout a
  period of duration t.
- **Availability**
  The availability of a system is the probability that the system will be functioning correctly at
  any given time.
  if a system is unavailable it is not delivering the specified system services.
- **Safety**
  Safety is a property of a system that will not endanger human life or the environment.
- **Security**
  Prevention of or protection against
  - access to information by unauthorized recipients or
  - intentional but unauthorized destruction or alteration of that information.

- dependability with respect to prevention of unauthorized access and/or handling of
  information
- The ability of the system to protect itself against accidental or deliberate intrusion
- Security is an essential prerequisite for availability, reliability and safety,If a system is a
  networked system and is insecure then it's unreliable.

3. Elements of Microcontroller:

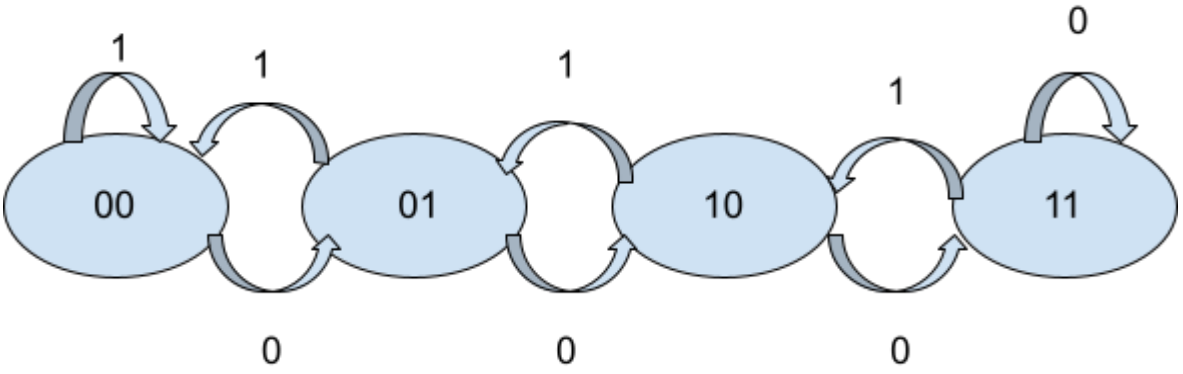


4. Typical processors for microcontrollers:

- **ARM Based Processors:**
  - ARMv7-M (Cortex-M3): 2^{32}, 32-bit RISC, 1,25 DMIPS/MHZ
  - ARMv8-A (Apple's iPhone/ Cortex-A): 64 bit, 1,5-2,5 GHZ

Cortex Families: Cortex-A (for complex computations), Cortex-M(low-cost embedded market), Cortex-R (High performance)

5. 2-bit predictor:



E (Event): 1 (taken)/ 0 (not taken)

State 00 & 11: Strongly predicted

State 01 & 10: Weakly predicted

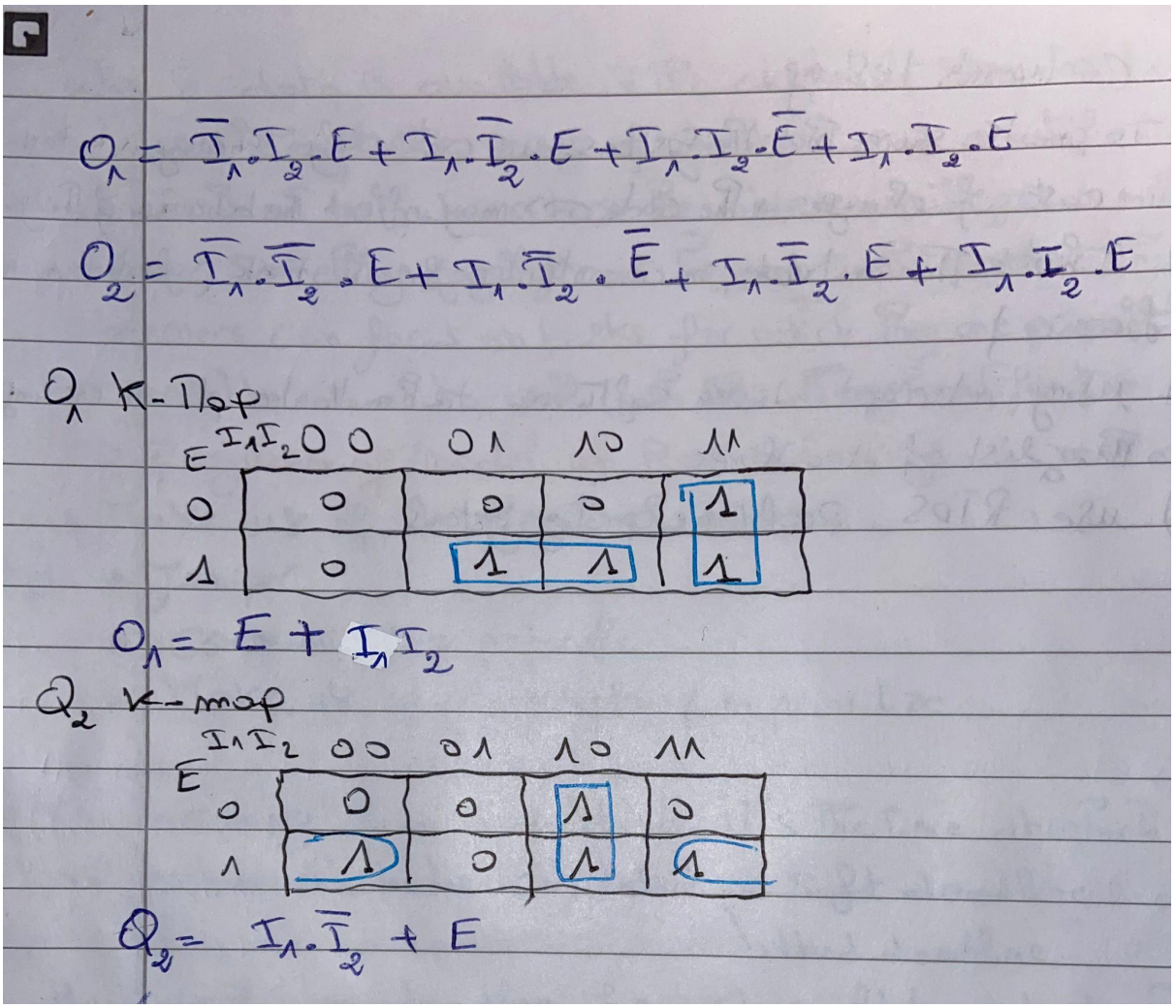
I1	I2	E	O1	O2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

- **C program:**

```
#include <stdio.h>
#define stronglyTaken 100
#define weaklytaken 101
#define weaklyNotTaken 110
#define stronglyNotTaken 111
#define taken 1
#define notTaken 0

int event, answer;

int state = stronglyTaken;
int main ()
{
    int event, answer;
  for (;;)
       {
          printf("choose event: 0 for taken and 1 for not taken \n\n", event, taken,
notTaken);
          scanf("%d", &event);
          switch (state)
          {
          case stronglyTaken:
             if (event == taken)

                state = stronglyTaken;
             else
               if (event == notTaken)
                   state = weaklytaken;

             break;
              case weaklytaken:
             if (event == taken)

             else
                if (event == notTaken)
                   state = weaklyNotTaken;

             break;
              case weaklyNotTaken:
             if (event == taken)

               state = weaklytaken;
             else
                if (event == notTaken)
                   state = stronglyNotTaken;
```

```
                break;
                 case stronglyNotTaken:
                if (event == taken)

                else
                  if (event == notTaken)
                      state = stronglyNotTaken;

                break;

            default:
                break;
            }
        printf (" The new state is = %d \t", state);
        printf ("\n\n");
        }

return 0;
}
```

- **Circuit:**



03:

Waterfall Model:

1. **Requirements:**

   The KOA team needs to meet the customer in order define the different requirements of the project and document them in a requirement specification document.

2. **Program Design:**

   Study the requirement specifications and prepare a design of the system. During the phase KOA team will define the overall architecture of the system.

3. **Implementation:**

   By using the output of the previous step the team will divide the system into small programs. Each developed unit will be tested for its functionality.

4. **Integration and Testing:**

   Integrate all the developed units into a system, then test the entire system to detect failures if they exist.

5. **Deployment of system:**

   Once the entire testing process is done (function/ non-functional testing). The system is deployed in the other company's environment.

6. **Maintenance:**

   Enhance the system for better versions and fix issues once they appear. These changes need to be delivered to the customer.

V model:

1. **Requirements Analysis:**

   Define the exact requirements and prepare the acceptance test design planning.

2. **System Design:**

   Design the complete system.

3. **Architecture Design:**

   Understand and design the architectural specifications. Break down the system design into modules according to the different functionalities (High Level Design).

4. **Module Design:**

   Specify the detailed internal design for all the modules of the system (Low Level Design).

5. **Coding:**

Develop the different modules after choosing the suitable programming language.

6. **Unit Testing:**

   Test at the code level and eliminate bugs.

7. **Integration Testing:**

   Test the coexistence and communication of the internal modules within the system.

8. **System Testing:**

   Test the entire system functionality and the communication of the system under development with external systems.

9. **Acceptance Testing:**

   Test the product in the other company's environment.

Additional phases of V model:

- **Architecture Design**
- **Module Design**
- **Unit testing**
- **Integration Testing**
- **Acceptance Testing**

The advantage of V model:

In the V model defects are identified during the early stages because the development cycle is directly associated with a testing phase.

Process Models Assessment:

Model	Size of team	Complexity of the project	Known Requirements	Change Of Requirement	Time to market	Knowledge of IT	Average number of iteration
Waterfall	**Small -Mid**	**Low-Mid**	**Known**	**Stable**	**Fast -Mid**	**No**	**0**
V model	**Mid**	**Mid**	**Known**	**Stable**	**Mid**	**No**	**0**
Agile	**Small**	**Low-Mid**	**------**	**Unstable**	**Fast**	**Yes**	**Many**
RUP	**Big**	**High**	**Unknown**	**Unstable**	**Slow**	**No**	**Many**
Spiral model	**Big**	**High**	**Unknown**	**Unstable**	**Slow**	**No**	**Many**

04:

Interrupt:

Arduino supports different Interrupts.

The function is: attachInterrupt(). The first parameter is the interrupt number using: digitalPinToInterrupt(pin).

- Syntax of Interrupt:
  attachInterrupt(digitalpinToInterrupt(pin), ISR, mode): recommended.
  or
  attachInterrupt(pin, ISR, mode)
- Parameters:
    - Interrupt: the number of the interrupt. Allowed data types: int
    - pin:the Arduino pin number.
    - ISR: to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.
    - Mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:
        - LOW to trigger the interrupt whenever the pin is low,
        - CHANGE to trigger the interrupt whenever the pin changes value
        - RISING to trigger when the pin goes from low to high,
        - FALLING for when the pin goes from high to low.
          The Due, Zero and MKR1000 boards allow also:
        - HIGH to trigger the interrupt whenever the pin is high
- Implementation:
  https://www.tinkercad.com/things/dBtfgvA2hiM