Microprocessor – Microcontroller

W05 – Creating an Embedded Operating System

Nguyen Tran Huu Nguyen
D: Computer Engineering
E: nthnguyen@hcmut.edu.vn

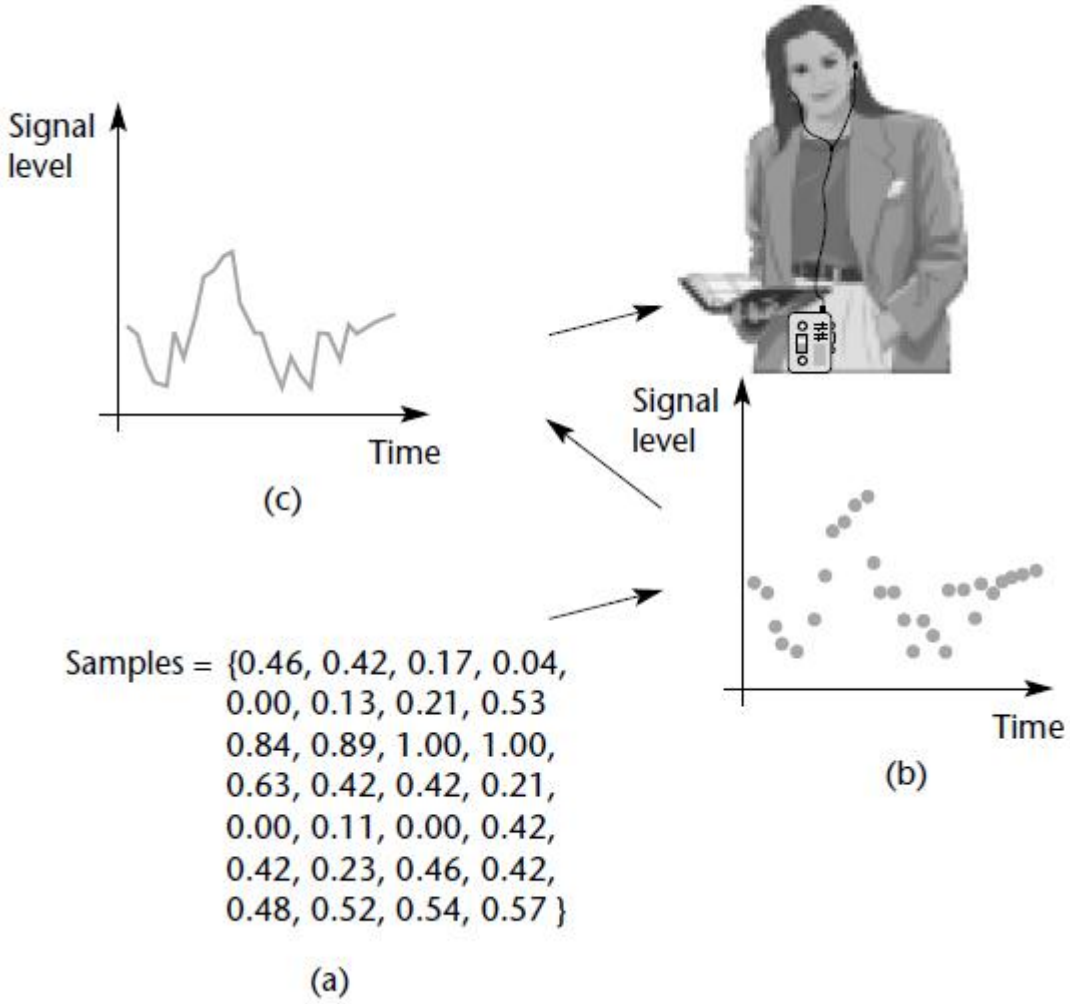
Super loop (round-robin) Architecture

```
void main(void)
{
   // Prepare run function X
   X_Init();
   // 'for ever' (Super Loop)
   while(1) {
      X(); // Run function X
   }
}
```

- it is easy to understand,
- it consumes virtually no system memory or CPU resources.
- it is very difficult to execute function X() at precise intervals of time

Consider an embedded application: an audio guide

- Such applications are used in museums and galleries to describe the numbered exhibits to visitors.



(a) The raw speech data, stored in ROM.
(b) The result of playing these data through a digital-to-analog converter, generating one sample every 0.2 ms.
(c) The (low-pass) filtered and amplified version of the converter output signal

Example: An audio guide

- The guide can typically be used in two ways.
  - First, the visitor can select **'auto pilot':** this will then cause the guide to describe a sequence of exhibits in order, in the following manner:
    - Item 345 was painted by Selvio Guaranteen early in the 16th century. At this time, Guaranteen, who is generally known as a member of the Slafordic School, was …
    - Now turn to your left, and locate Item 346, a small painting which was until recently also thought to have been painted by Guarateen but which is now … .
  - Second, the visitor can use the device **in 'manual' mode**.
    - the user will be free to wander at leisure around the exhibition and, when he or she finds an item of interest, they will type in the exhibit number on the electronic guide: they will then hear the relevant commentary.

Example: An audio guide

- No matter what technique we use, the basic processing required in this application will be the same:
  - we need to generate a long stream of speech from a store of data in memory.
  - This will typically involve using a digital-to-analog converter to generate an analog signal at a rate of at least 5000 samples per second.
- The need to call the same function repeatedly, at precise intervals, is by no means restricted to this museum system (or similar systems such as MP3 players).

Other examples

- The current speed of the vehicle must be measured at 0.5 second intervals.
- The display must be refreshed 40 times every second.
- The calculated new throttle setting must be applied every 0.5 seconds.
- A time-frequency transform must be performed 20 times every second.
- The engine vibration data must be sampled 1000 times per second.
- The frequency-domain data must be classified 20 times every second.
- The keypad must be scanned every 200 ms.
- The master (control) node must communicate with all other nodes (sensor nodes and sounder nodes) once per second.
- The new throttle setting must be calculated every 0.5 seconds.
- The sensors must be sampled once per second.

- In practice, many embedded systems must be able to support this type of ‘periodic function’:
  - that is, activities that are performed repeatedly, every millisecond or every ten milliseconds.
- In most cases, the process must take place at precisely the specified interval if the device is to operate as required.
- In the case of the museum guide, for example, imprecise function timing will result in unpleasant distortions to the frequency components in the signal.

Using the Super Loop architecture to execute functions at regular intervals

```
void main(void)
{
   // Prepare run function X
   X_Init();
   // 'for ever' (Super Loop)
   while(1) {
      X(); // Run function X
10ms
      Delay_ms(50);
   }
}
```

X() runs in every 60ms

It is good if

- We know the precise duration of function X(), and,
- This duration never varies.

In practical applications, determining the precise function duration is rarely straightforward.

Suppose we have a very simple function that does not interact with the outside world but, instead, performs some internal calculations.

Even under these rather restricted circumstances,

- changes to compiler optimization settings
- even changes to an apparently unrelated part of the program

can alter the speed at which the function executes. This can make fine-tuning the timing very tedious and error-prone.

The second condition is even more problematic. Often in an embedded system functions will be required to interact with the outside world in a complex way.

- In these circumstances the function duration will vary according to outside activities in a manner over which the programmer has very little control.

Timer-based Interrupt (the core of an embedded OS)

```
void FSM();
void main(void){
    // initialize the device
    System_Initialization();
    while (1) {
                SLEEP()
    }
}
/**
  * @brief  This function handles TIM interrupt request.
  * @param  None
  * @retval None   */
void TIM3_IRQHandler(void){
     HAL_TIM_IRQHandler(&TimHandle);
}
```

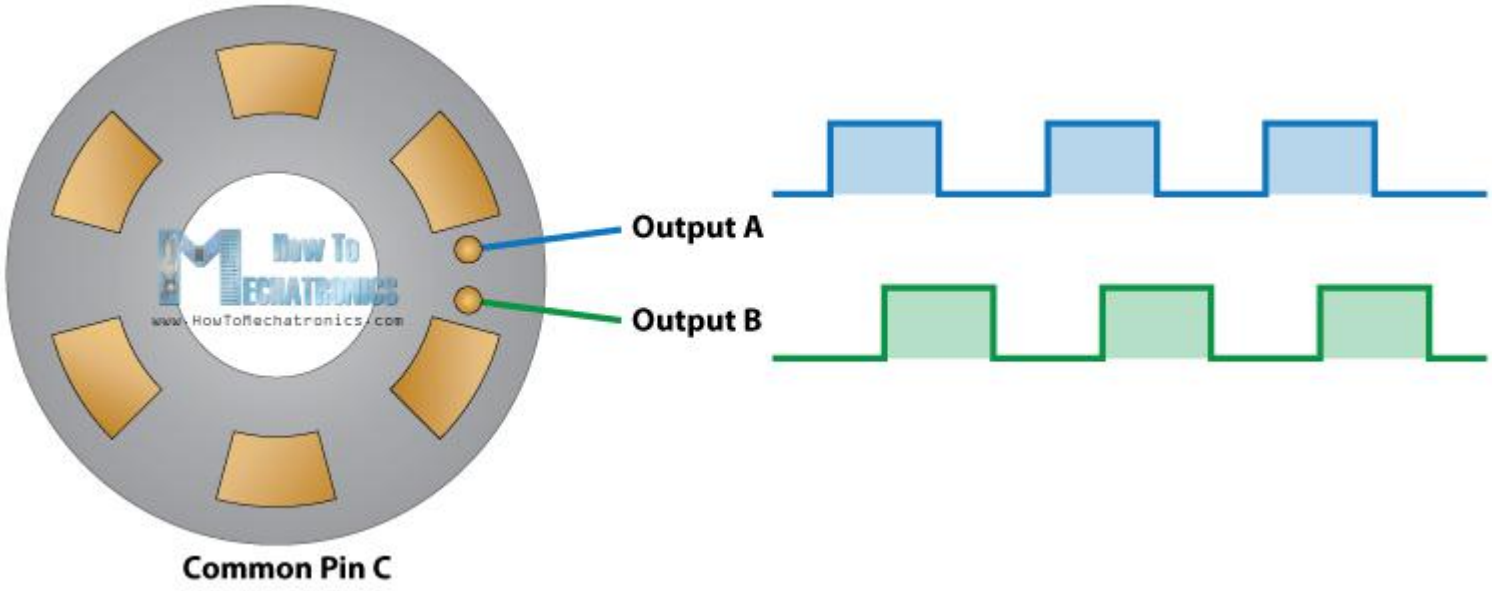
```
/**
  * @brief  Period elapsed callback in non blocking mode
  * @param  htim : TIM handle
  * @retval None  */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
        Timer1_ISR();
}
```

Exercise 1

- Calculate the speed of a rotary encoder
- A rotary encoder is a type of position sensor which is used for determining the angular position of a rotating shaft.

Rotary Encoder

- The encoder has a disk with evenly spaced contact zones that are connected to the common pin C and two other separate contact pins A and B.



How Rotary Encoder Works

- When the disk starts rotating, the pins A and B starts making contact with the common pin and the two square wave output signals are generated accordingly.
- If we just count the pulses of the signal, any of the two outputs can be used for determining the rotated position.
- If we want to determine the rotation direction, we need to consider both signals at the same time.

- The two output signals are displaced at 90 degrees out of phase from each other.
- If the encoder is rotating clockwise the output A will be ahead of output B.
  - At the time the signal changes, from High to Low or from Low to High, the two output signals have opposite values.
- if the encoder is rotating counterclockwise, the output signals have equal values.