BEÁGYAZOTT RENDSZEREK (C)

1. forduló

BOSCH

A kategória támogatója: Robert Bosch Kft.

Ismertető a feladathoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

- MINDEN kérdésre **van helyes válasz**.
- Olyan kérdés **NINCS**, amire az összes válasz helyes, ha mégis az összes választ bejelölöd, arra a feladatra automatikusan 0 pont jár.
- A **radio button-os kérdésekre egy helyes válasz van**.
- **Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN** beküldi azt az addig megjelölt válaszokkal.
- Azokat a feladatlapokat, amelyekhez **csatolmány** tartozik, javasoljuk **NEM mobilon** elindítani, erre az érintett feladatlapok előtt külön felhívjuk a figyelmet.
- Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.
- **Helyezéseket a 4. forduló után mutatunk**, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.
- **Badge-ket** szintén a 4.forduló után kapsz majd először.
- Ha egyszerre több böngészőből, több ablakban vagy több eszközről megnyitod ugyanazt a feladatlapot, **nem tudjuk vállalni** az adatmentéssel kapcsolatban esetlegesen felmerülő anomáliákért a felelősséget!
- A hét forduló során az egyes kategóriákban (de nem feltétlenül mindegyikben) **könnyű-közepes-nehéz kérdésekkel** egyaránt találkozhatsz majd.

Jó versenyzést kívánunk!

1.forduló

Gipsz Szabolcsnak van egy robotporszívója, mely meghibásodott: a porszívó szoftveresen működik, az okostelefonnal kommunikál, de nem mozdul semerre. Szabolcs a motorok vezérlésének hibájára gyanakszik. Sajnos a porszívó már nem garanciális, így maga kezdi el megvizsgálni a probléma okát és próbálkozik a javítással.

Szabolcs azt tapasztalja, hogy terhelés nélkül a kerekek egy picit megmozdulnak, amikor a robot elindulna, de rögtön meg is állnak. Arra gyanakszik, hogy vagy a teljesítményelektronikában lehet a hiba, vagy az odometriáért felelős enkóderekből nem érkezik megfelelő jel a vezérlőhöz.

Nemrég kapott egy barátjától egy Arduino alapú logikai analizátort/oszcilloszkópot, így úgy dönt, hogy kipróbálja a műszert, egyúttal az odometria vizsgálatával kezdi a hibakeresést.

A megoldásban a következő adatlapok lesznek a segítségedre:

STM32F091RC adatlap: https://www.st.com/resource/en/datasheet/stm32f091rc.pdf

NUCLEO-F091RC adatlap: https://www.st.com/resource/en/data_brief/nucleo-f091rc.pdf

HAL API dokumentáció: https://www.st.com/resource/en/user_manual/dm00122015-description-of-stm32f0-hal-and-lowlayer-drivers-stmicroelectronics.pdf

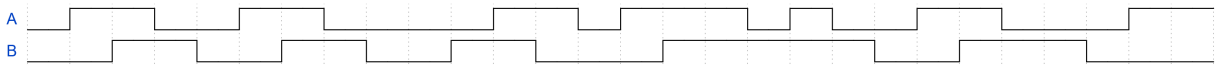
AEDR-8300 enkóder adatlap: https://docs.broadcom.com/doc/AV02-3572EN

Felhasznált idő: 00:00/20:00 **Elért pontszám:** 0/23

1. feladat 0/4 pont

Az ábrán egy logikai analizátor kijelzőjén megjelenített jelalakot látsz. Az *A* és *B* csatorna egy inkrementális enkóder kvadratúra kimenetein mért értékeket jeleníti meg.

Hány osztásnyival mozdult el az enkóder tárcsája a kiinduló állapothoz képest (számmal add meg)? *(Megjegyzés: mindkét kimeneten a fel- és lefutó élek számlálásával a felbontás növelhető, de itt most csak a tárcsán lévő osztások számával mért abszolút elmozdulásra vagyunk kíváncsiak.)*



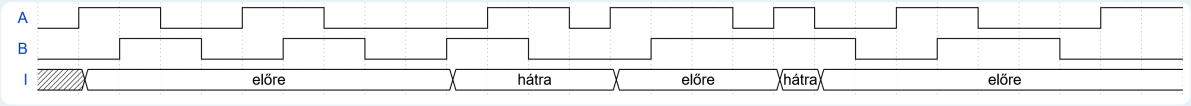
Válasz

A helyes válasz:

3

Magyarázat

A tárcsát két osztásnyit előre, majd egy osztásnyit vissza, majd előre, picit vissza, végül előre forgattuk, ahogy az alábbi ábra is mutatja.



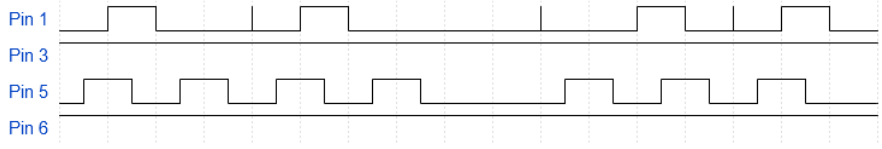
Ha az *A* csatornán számolod a felfutó éleket, közben az irány meghatározásához a *B* csatornán a jel szintjét, akkor 5 előre és 2 visszalépést számolhatsz össze.

Ha a *B* csatornán számolod a felfutó éleket, közben az irány meghatározásához az *A* csatornán a jel szintjét, akkor 4 előre és 1 visszalépést számolhatsz össze.

2. feladat 0/4 pont

Szabolcs az enkódert tartó kis NYÁK-ra szitázva megtalálta az enkóder típusjelzését. E szerint a porszívó kerekeinél egy-egy Avago AEDR-8300 inkrementális enkóder dolgozik. Úgy tűnik, ez az enkóder modul egy cserélhető egység, ami az enkóderen kívül pár ellenállást, kapacitást és egy csatlakozót tartalmaz.

Szabolcs felderítette az enkóder kivezetéseit melyekre csatlakoztatta logikai analizátorát, majd egyik és másik irányba is elforgatta a kerekeket. A jobb oldali keréknél ránézésre mindent rendben talált, a bal oldalinál azonban az alábbi jelszinteket mérte.



Segíts neki eldönteni, hogy mi lehet a hiba oka!

Válasz

- ○ Az emitter LED túláram miatt tönkremehetett.
- ○ Az *A* csatorna huzalozása kontakthibás.
- ● A B csatorna huzalozása kontakthibás.
  Ez a válasz helyes, de nem jelölted meg.
- ○ Az érzékelő kimozdult az optimális pozícióból, így a két csatorna érzékelőterületei közötti fáziseltolás mértéke jelentősen lecsökkent.

Magyarázat

Az enkóder adatlapjáról kikereshető, hogy az *A* és *B* csatorna a tokozás 5-ös és 1-es lábára van kivezetve.

A logikai analizátor által megjeleített jelekből látható, hogy a *B* csatorna (Pin 1) olyankor is alacsony szinten maradt, amikor magas szintet várnánk. Erre persze magyarázat lehetne, ha „remegő kézzel" forgattuk volna meg a kereket, de a jelenség forgatás közben, ráadásul többször és nem a forgatás elindításánál jelentkezett. Esélyes, hogy a Pin 1 huzalozása kontakthibás vagy maga a szenzor sérült. Mivel időnként magas jelet is mértünk, utóbbi nem valószínű, továbbá apró tüskéket is megfigyelhetünk ezen a csatornán, melyek valószínűleg a mechanikai rezgések hatására a hibás kontaktus pillanatnyi záródásából következnek.

3. feladat 0/15 pont

A csatlakozásokat átforrasztotta, de a hiba nem szűnt meg, Szabolcs vett egy új enkóder modult. Mielőtt beépítené a porszívóba, szeretné tesztelni. Ehhez pont van nála kölcsönben egy STM32 Nucleo-F091RC fejlesztőkártya. Holnap már vissza kell adnia, de most erre a célra még pont jól jön.

A vezérlő PA6 és PA7 lábaira köti az enkóder A és B kimenetét.

Melyik kódrészlet valósítja meg helyesen az enkóder kezelését?

Válaszok

- ☐ A válasz:

```c
#include "main.h"
#include "stdio.h"

UART_HandleTypeDef huart2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();

  GPIO_PinState pa6_prevstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6);
  long position = 0;

  while (1)
  {
      GPIO_PinState pa6_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6);
      GPIO_PinState pa7_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_7);
      if(pa6_currstate != pa6_prevstate) {
          if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_RESET)||
             (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_SET))
              position++;
          if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_SET)||
             (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_RESET))
              position--;
          pa6_prevstate = pa6_currstate;
          char msg[24];
          snprintf(msg, 24, "Position: %ld\r\n", position);
          HAL_UART_Transmit(&huart2, msg, sizeof(msg), HAL_UART_TIMEOUT_VALUE);
      }
      HAL_Delay(10);
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PA6 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}
```

B válasz:

```c
#include "main.h"
#include "stdio.h"

UART_HandleTypeDef huart2;
long position;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();

  position = 0;
  long last_printed_position = -1;

  while (1)
  {
      if(position != last_printed_position) {
          char msg[24];
          snprintf(msg, 24, "Position: %ld\r\n", position);
          HAL_UART_Transmit(&huart2, msg, sizeof(msg), HAL_UART_TIMEOUT_VALUE);
          last_printed_position = position;
      }
      HAL_Delay(10);
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI4_15_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI4_15_IRQn);

}

void HAL_GPIO_EXTI_Callback( uint16_t GPIO_Pin)
{
```

```
    if(GPIO_Pin == GPIO_PIN_6) {
        GPIO_PinState pa6_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6);
        GPIO_PinState pa7_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_7);
        if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_RESET)||
           (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_SET))
            position++;
        if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_SET)||
           (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_RESET))
            position--;
    }
}
```

Ez a válasz helyes, de nem jelölted meg.

C válasz:

```c
#include "main.h"
#include "stdio.h"

UART_HandleTypeDef huart2;
long position;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();

  position = 0;
  long last_printed_position = -1;

  while (1)
  {
      if(position != last_printed_position) {
          char msg[24];
          snprintf(msg, 24, "Position: %ld\r\n", position);
          HAL_UART_Transmit(&huart2, msg, sizeof(msg), HAL_UART_TIMEOUT_VALUE);
          last_printed_position = position;
      }
      HAL_Delay(10);
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PA6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI4_15_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI4_15_IRQn);

}

void HAL_GPIO_EXTI_Callback( uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_6 || GPIO_Pin == GPIO_PIN_7) {
```

```
    if(GPIO_Pin == GPIO_PIN_6 || GPIO_Pin == GPIO_PIN_7) {
        GPIO_PinState pa6_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6);
        GPIO_PinState pa7_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_7);
        if((pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_RESET)||
           (pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_SET))
            position++;
        if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_SET)||
           (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_RESET))
            position--;
    }
}
```

☑ D válasz:

```c
#include "main.h"
#include "stdio.h"

UART_HandleTypeDef huart2;
long position;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();

  position = 0;
  long last_printed_position = -1;

  while (1)
  {
      if(position != last_printed_position) {
          char msg[24];
          snprintf(msg, 24, "Position: %ld\r\n", position);
          HAL_UART_Transmit(&huart2, msg, sizeof(msg), HAL_UART_TIMEOUT_VALUE);
          last_printed_position = position;
      }
      HAL_Delay(10);
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PA6 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI4_15_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(EXTI4_15_IRQn);

}

void HAL_GPIO_EXTI_Callback( uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_6 || GPIO_Pin == GPIO_PIN_7) {
        GPIO_PinState pa6_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6);
        GPIO_PinState pa7_currstate = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_7);
        if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_RESET)||
           (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_SET))
            position++;
```

```
            position++;
        if((pa6_currstate == GPIO_PIN_SET && pa7_currstate == GPIO_PIN_SET)||
           (pa6_currstate == GPIO_PIN_RESET && pa7_currstate == GPIO_PIN_RESET))
            position--;
    }
}
```

E válasz:

```c
#include "main.h"
#include "stdio.h"

TIM_HandleTypeDef htim3;
UART_HandleTypeDef huart2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM3_Init(void);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM3_Init();

  long position = 0;
  long last_printed_position = -1;

  while (1)
  {
      position = htim3.Instance->CNT;
      if(position != last_printed_position) {
          char msg[24];
          snprintf(msg, 24, "Position: %ld\r\n", position);
          HAL_UART_Transmit(&huart2, msg, sizeof(msg), HAL_UART_TIMEOUT_VALUE);
          last_printed_position = position;
      }
      HAL_Delay(10);
  }
}

static void MX_TIM3_Init(void)
{
  TIM_Encoder_InitTypeDef sConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 0;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 65535;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  sConfig.EncoderMode = TIM_ENCODERMODE_TI1;
  sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC1Filter = 0;
  sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC2Filter = 0;
  if (HAL_TIM_Encoder_Init(&htim3, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

}
```

Ez a válasz helyes, de nem jelölted meg.

Magyarázat

A 10 ms-os pergésmentesítésre ennél az enkódernél nincsen szükség, sőt, kifejezetten kerülendő, hiszen pont, hogy nem szeretnénk elszalasztani jelátmeneteket. Polling esetén – különösen, ha a vezérlő egyéb feladatokat is végez – fennáll a veszélye, hogy jelszint-átmeneteket nem veszünk észre, ezért ilyen alkalmazásoknál ez a módszer kerülendő.

A második és a negyedik megoldás is jó, igaz a negyedik precízebb detektálást végez.

A harmadik megoldás majdnem jó, de a megszakításkezelő rutinban a számláló növeléséhez és csökkentéséhez tartozó feltételek logikailag „véletlenül” azonosak, valamint a PA6-os bemeneten csak felfutó, a PA7-esen csak lefutó élre kértünk megszakítást, ami hibátlan logika mellett is „egyenetlen” számláláshoz vezetne.

Az ötödik megoldás a legcélszerűbb és legszebb: ha van hardveres támogatás egy funkcióra a vezérlőben, érdemes azt használni a vezérlő képességeinek és kapacitásának minél jobb kihasználása érdekében.