LAB NOTE

Subject: Hardware/Software Interfacing

Lab 5: Stepper Motor

Student: Minh Quan Tran

Oct 10th, 2024

Table of Contents

1. Objectives 4

2. Problems and Solutions 5

2.1 Problems 5

2.2 Solutions 5

3. Software Design 6

3.1 List of function 6

3.2 While loop 8

TABLE OF FIGURES

Figure 1-1: Basic outcome 4
Figure 1-2: Intermediatet outcome 4
Figure 1-3: Advance outcome 4
Figure 1-4: Maximum outcome 5

1. Objectives

- Using STM32F411 board and STM32CubeIDE in Windows, create code to
  • Write code for the STM32F411 to configure and use Timers.

Basic:

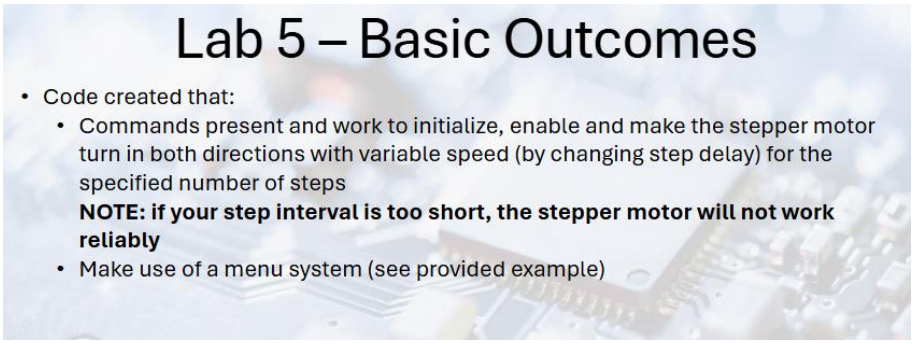

Figure 1-1: Basic outcome

- Intermediate:

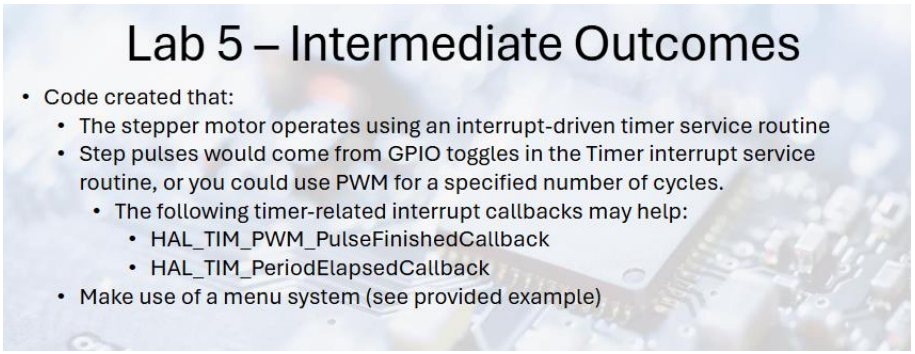

Figure 1-2: Intermediatet outcome

- Advance:

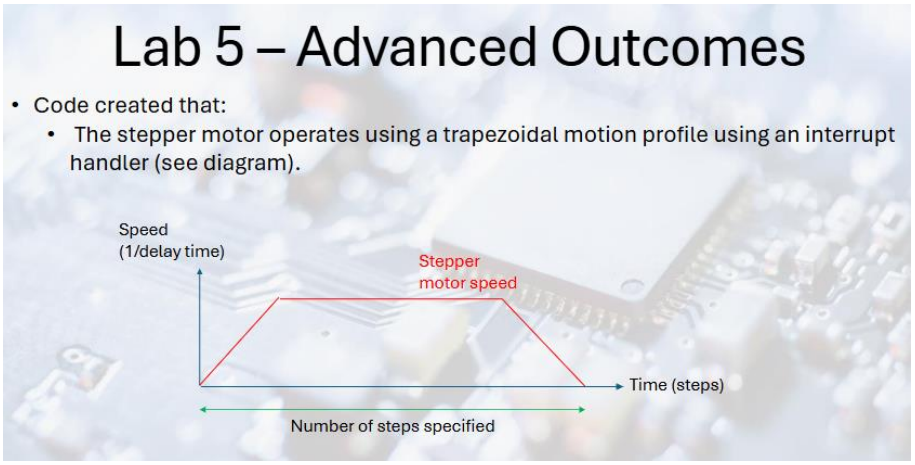

Figure 1-3: Advance outcome

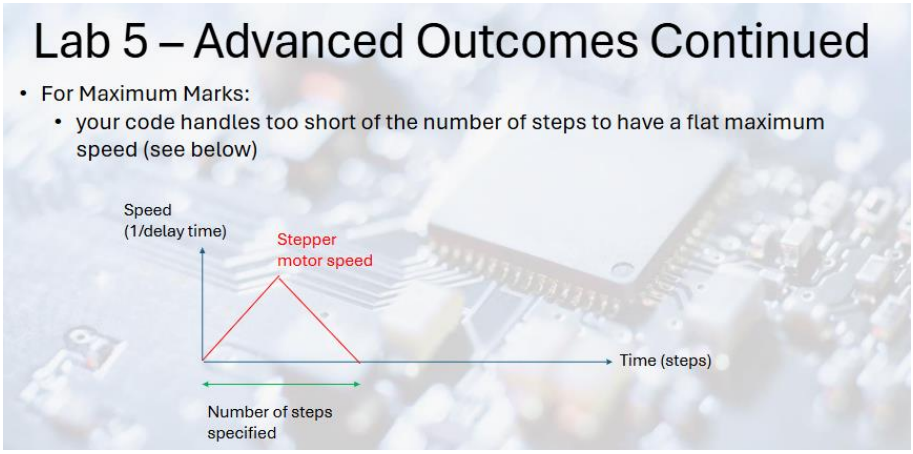

Figure 1-4: Maximum outcome

2. Problems and Solutions

2.1 Problems

- No problem

2.2 Solutions

3. Software Design

3.1 List of function

- This function is to create a delay in micro second by counting the number of ticks has passed using CNT registers.

```
void microDelay(uint32_t usDelay)
{
    // Get the current timer counter value
    uint32_t startTime = __HAL_TIM_GET_COUNTER(&htim1);
    uint32_t ticks = usDelay - 1; // 1 ticks = 1 us

    // Poll the CNT register until the specified number of ticks has passed
    while ((__HAL_TIM_GET_COUNTER(&htim1) - startTime) < ticks)
    {
        // Wait until the timer reaches the required ticks
    }
}
```

- This function is used to toggle LED whenever the timer overflow

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM11)
    {
        HAL_GPIO_TogglePin(GPIOB, Step_Pin);
        static uint32_t stepCount = 0;
        stepCount++;
#ifdef ADVANCE
        if (stepCount > 2)
        {
            HAL_TIM_Base_Stop_IT(&htim11);
            stepCount = 0;
            runFlag = DONE;
        }
#else
        if (stepCount > abs(numberOfSteps) * 2)
        {
            HAL_TIM_Base_Stop_IT(&htim11);
            stepCount = 0;
            runFlag = DONE;
        }
#endif
    }
}
```

- This function is used to initialize the motor

```
void stepInit(void)
{
      HAL_GPIO_WritePin(GPIOC,Reset_Pin,GPIO_PIN_SET); // High - Normal operation, Low - Reset
      HAL_GPIO_WritePin(GPIOC,PS_Pin,GPIO_PIN_SET);    // High - Operation mode, Low - Standby mode
      HAL_GPIO_WritePin(GPIOA,VRef_Pin,GPIO_PIN_SET);  // High - 100% current, Low - 0 current
      printf("Inital Motor \r\n");
}
```

- This function is used to enable/disable the motor

3. Result

```
void stepEnable(bool enable)
{
	if(!enable)
	{
		HAL_GPIO_WritePin(GPIOC,OE_Pin,GPIO_PIN_SET);
		printf("Disable Motor\r\n");
	}
	else
	{
		HAL_GPIO_WritePin(GPIOC,OE_Pin,GPIO_PIN_RESET);
		printf("Enable Motor\r\n");
	}
	return;
}
```

- This function is used to control the number of steps and how long will a step will run in the motor

```
void stepSteps(int32_t numberOfSteps, uint32_t usDelay)
{
#ifdef USE_INTERRUPT
   runFlag = IN_PROCESS;

   htim11.Init.Period = usDelay / 2 - 1;
   if (HAL_TIM_Base_Init(&htim11) != HAL_OK)
   {
		Error_Handler();
   }
   HAL_TIM_Base_Start_IT(&htim11);

   //Wait until finish step
   while(runFlag == IN_PROCESS);

#else // Basic (No interrupt needed)

   // Tell which direction the motor is running
   if (numberOfSteps > 0)
   {
		HAL_GPIO_WritePin(GPIOC,FR_Pin,GPIO_PIN_SET);
   }
   else
   {
		HAL_GPIO_WritePin(GPIOC,FR_Pin,GPIO_PIN_RESET);
   }

   printf("Stepping Motor\r\n");
   for (int i = 0; i < abs(numberOfSteps); i++)
   {
		HAL_GPIO_WritePin(GPIOB,Step_Pin,GPIO_PIN_SET);
		microDelay(usDelay / 2);
		HAL_GPIO_WritePin(GPIOB,Step_Pin,GPIO_PIN_RESET);
		microDelay(usDelay / 2);
   }
   printf("Finish Stepping\r\n");
#endif

}
```

3. Result

- This function is used to run the motor using the trapezoidal motion profile

```
#ifdef ADVANCE
void runTrapezoidalMotion(int32_t numberOfSteps, uint32_t usDelay)
{
   // Tell which direction the motor is running
   if (numberOfSteps > 0)
   {
         HAL_GPIO_WritePin(GPIOC,FR_Pin,GPIO_PIN_SET);
   }
   else
   {
         HAL_GPIO_WritePin(GPIOC,FR_Pin,GPIO_PIN_RESET);
   }

   numberOfSteps = abs(numberOfSteps);

    // Ensure number of rampSteps is not greater than half of the total number of
steps
    int32_t rampSteps = MIN(RAMP_STEP, numberOfSteps / 2);

    // Middle steps at max speed
    int32_t middleSteps = numberOfSteps - 2 * rampSteps;

    printf("Stepping Motor\r\n");
    // Acceleration Phase
    for (int32_t i = 0; i < rampSteps; i++)
    {
        // Linearly interpolate delay between max speed and min speed
        uint32_t currentDelay = MAX_DELAY - (MAX_DELAY - usDelay) * i / rampSteps;
        stepSteps(1, currentDelay);
    }

    // Constant Speed Phase
    for (int32_t i = 0; i < middleSteps; i++)
    {
        stepSteps(1, usDelay);  //Intended speed
    }

    // Deceleration Phase
    for (int32_t i = 0; i < rampSteps; i++)
    {
        // Linearly interpolate delay between minUsDelay and maxUsDelay
        uint32_t currentDelay = usDelay + (MAX_DELAY - usDelay) * i / rampSteps;
        stepSteps(1, currentDelay);
    }

    printf("Finish Stepping\r\n");
}
#endif
```

3.2 While loop

```
 while (1)
 {
    if(doPrompt != 0)
    {
          doPrompt = 0;
          printf(PROMPT);
    }


    switch(GetCharFromUART2())
```

3. Result

```
      {
      case '0':
            enableFlag = !enableFlag;
            stepEnable(enableFlag);
            break;
      case '1':
            numberOfSteps += 400; // Rotate an addition 90o
            printf("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay );
            break;
      case '2':
            numberOfSteps -= 400; // Reduce 90 degree
            printf("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay );
            break;
      case '3':
            if (delay < MAX_DELAY)
            {
                  delay += 50;
                  printf("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay
);
            }
            else
            {
                  printf("Delay can't be higher than %duS\r\n", MAX_DELAY);
            }
            break;
      case '4':
            if (delay > MIN_DELAY)
            {
                  delay -= 50;
                  printf("Set steps = %ld, Set delay = %u \r\n",numberOfSteps, delay
);
            }
            else
            {
                  printf("Delay can't be lower than %duS\r\n", MIN_DELAY);
            }
            break;
      case '5':
#ifdef ADVANCE
            runTrapezoidalMotion(numberOfSteps,delay);
#else
            stepSteps(numberOfSteps,delay);
            printf("RUNNING\r\n");
#endif
            break;
      case TIMEOUT_ERROR:
            printf("Prompt timeout\r\n");
            break;
      default:
            break;
      }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
```

REFERENCES