HX711 with Load cell

Development board- STM32F303VCT6

Connections of HX711 with STM32-

* GND ---- GND
* VCC ---- 3V
* DT ---- PD5(GPIO\_INPUT)(any gpio pin)
* SCK ---- PD6(GPIO\_OUTPUT)(any gpio pin)

Connections of HX711 with Load cell-

* E+ ---- RED Wire
* E- ---- BLACK Wire
* A- ---- WHITE Wire
* A+ ---- GREEN Wire

Library file (load.h)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #include "main.h"  #include "tim.h"  #define \_HX711\_DELAY\_US\_LOOP 4  //#define Weight\_Lode  //#define HX711\_Buffer  **void** **hx711\_delay\_us**(**void**);  **unsigned** **long** **hx711\_read**(**void**);  **void** **Get\_Weight**(**void**); |

* Include that load.h into main.h
* In load.h we define the function which we are going to use in our program.
* Load.c is having the code for reading data and calibration code

Load.c

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68 | #include "load.h" //you have to add this header file in main.h  **unsigned** **long** Weight\_No\_Lode = **32279** ; //it is the value of count which we recieve in debug mode without load  **unsigned** **long** HX711\_Buffer; // the avg value of 10 values of count  **unsigned** **long** Weight\_Lode; //it is the value of count which we recieve in debug mode with load  **unsigned** **long** Weight\_Real; //real weight after calibration(we are calibrating with known load)  **unsigned** **long** Kp\_Weight = **43**; //it is calibrating by the known weight of 1kg,2kg,3kg take the avg value  **uint8\_t** i;  **uint32\_t** count = **0**;  **void** **delay\_us** (**uint16\_t** us) //it is for generating microsecond delay using with timer  {  \_\_HAL\_TIM\_SET\_COUNTER(&htim1,**0**); // set the counter value a 0  **while** (\_\_HAL\_TIM\_GET\_COUNTER(&htim1) < us); // wait for the counter to reach the us input in the parameter  }  **unsigned** **long** **hx711\_read**() // function for read data from hx711  {  HAL\_GPIO\_WritePin(clock\_GPIO\_Port,clock\_Pin, GPIO\_PIN\_RESET); //make clock pin low first    HAL\_GPIO\_TogglePin(led1\_GPIO\_Port,led1\_Pin); //just for check    delay\_us(**1**); //using from microsecond delay function which we define in load.c  count = **0**; //SET count inhitially zero    **for**(i=**0**;i<**24**;i++) //it is a 24 bit ADC that's why we give a counting to 24  {  HAL\_GPIO\_WritePin(clock\_GPIO\_Port,clock\_Pin,GPIO\_PIN\_SET);    count = count<< **1**; //shift the count from 1 after the seting the clock high  delay\_us(**11**);    HAL\_GPIO\_WritePin(clock\_GPIO\_Port,clock\_Pin,GPIO\_PIN\_RESET);    **if**(HAL\_GPIO\_ReadPin(data\_GPIO\_Port,data\_Pin))  count++; //if it reads data from data pin then increse the count from  delay\_us(**6**);  }  **return** count;    }  //calibration part start from here  **void** **Get\_Weight**(**void**) // for calibration to read function weight  {  **for**(**uint8\_t** j =**1**; j<=**10**; j++) //to collect avg value of weight from hx711\_buffer, we take avg of 10 values  {  HX711\_Buffer += count; //add 10 values of hx711\_buffer  }    Weight\_Lode = HX711\_Buffer/**10**; //divide by 10 gor taking avg  HX711\_Buffer = **0**;    //Judgment is not empty  **if**(Weight\_Lode> Weight\_No\_Lode)  {  Weight\_Real = (Weight\_Lode-Weight\_No\_Lode)/Kp\_Weight;//formula for calibration  }  **else** **if**(Weight\_Lode <= Weight\_No\_Lode)  Weight\_Real = **0.0f**;    //The serial port will display when the pulling force reaches a certain threshold  //if(Weight\_Real>100.0f)  //printf("Current pull %u\r\n\r\n",Weight\_Real);  } |

Main.c

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205 | /\* USER CODE BEGIN Header \*/  /\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @file : main.c  \* @brief : Main program body  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* @attention  \*  \* Copyright (c) 2022 STMicroelectronics.  \* All rights reserved.  \*  \* This software is licensed under terms that can be found in the LICENSE file  \* in the root directory of this software component.  \* If no LICENSE file comes with this software, it is provided AS-IS.  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*/  /\* USER CODE END Header \*/  /\* Includes ------------------------------------------------------------------\*/  #include "main.h"  #include "i2c.h"  #include "spi.h"  #include "tim.h"  #include "gpio.h"  /\* Private includes ----------------------------------------------------------\*/  /\* USER CODE BEGIN Includes \*/  /\* USER CODE END Includes \*/  /\* Private typedef -----------------------------------------------------------\*/  /\* USER CODE BEGIN PTD \*/  /\* USER CODE END PTD \*/  /\* Private define ------------------------------------------------------------\*/  /\* USER CODE BEGIN PD \*/  /\* USER CODE END PD \*/  /\* Private macro -------------------------------------------------------------\*/  /\* USER CODE BEGIN PM \*/  /\* USER CODE END PM \*/  /\* Private variables ---------------------------------------------------------\*/  /\* USER CODE BEGIN PV \*/    /\* USER CODE END PV \*/  /\* Private function prototypes -----------------------------------------------\*/  **void** **SystemClock\_Config**(**void**);  /\* USER CODE BEGIN PFP \*/  /\* USER CODE END PFP \*/  /\* Private user code ---------------------------------------------------------\*/  /\* USER CODE BEGIN 0 \*/  /\* USER CODE END 0 \*/  /\*\*  \* @brief The application entry point.  \* @retval int  \*/  **int** **main**(**void**)  {  /\* USER CODE BEGIN 1 \*/  /\* USER CODE END 1 \*/  /\* MCU Configuration--------------------------------------------------------\*/  /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/  HAL\_Init();  /\* USER CODE BEGIN Init \*/  /\* USER CODE END Init \*/  /\* Configure the system clock \*/  SystemClock\_Config();  /\* USER CODE BEGIN SysInit \*/  /\* USER CODE END SysInit \*/  /\* Initialize all configured peripherals \*/  MX\_GPIO\_Init();  MX\_I2C1\_Init();  MX\_SPI1\_Init();  //MX\_USB\_PCD\_Init();  MX\_TIM1\_Init();  /\* USER CODE BEGIN 2 \*/    HAL\_TIM\_Base\_Start(&htim1); //start the timer to generate microsecond delay    /\* USER CODE END 2 \*/  /\* Infinite loop \*/  /\* USER CODE BEGIN WHILE \*/  **while** (**1**)  {  hx711\_read(); //it reads the function in load.c for get value from hx711  Get\_Weight(); //it reads the function in load.c for calibration of weight  HAL\_Delay(**1000**);    /\* USER CODE END WHILE \*/  /\* USER CODE BEGIN 3 \*/  }  /\* USER CODE END 3 \*/  }  /\*\*  \* @brief System Clock Configuration  \* @retval None  \*/  **void** **SystemClock\_Config**(**void**)  {  RCC\_OscInitTypeDef RCC\_OscInitStruct = {**0**};  RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {**0**};  RCC\_PeriphCLKInitTypeDef PeriphClkInit = {**0**};  /\*\* Initializes the RCC Oscillators according to the specified parameters  \* in the RCC\_OscInitTypeDef structure.  \*/  RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI|RCC\_OSCILLATORTYPE\_HSE;  RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;  RCC\_OscInitStruct.HSEPredivValue = RCC\_HSE\_PREDIV\_DIV1;  RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;  RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;  RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;  RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;  RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL9;  **if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)  {  Error\_Handler();  }  /\*\* Initializes the CPU, AHB and APB buses clocks  \*/  RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK  |RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;  RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;  RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;  RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV2;  RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;  **if** (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_2) != HAL\_OK)  {  Error\_Handler();  }  PeriphClkInit.PeriphClockSelection = RCC\_PERIPHCLK\_USB|RCC\_PERIPHCLK\_I2C1  |RCC\_PERIPHCLK\_TIM1;  PeriphClkInit.I2c1ClockSelection = RCC\_I2C1CLKSOURCE\_HSI;  PeriphClkInit.USBClockSelection = RCC\_USBCLKSOURCE\_PLL\_DIV1\_5;  PeriphClkInit.Tim1ClockSelection = RCC\_TIM1CLK\_HCLK;  **if** (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInit) != HAL\_OK)  {  Error\_Handler();  }  }  /\* USER CODE BEGIN 4 \*/  /\* USER CODE END 4 \*/  /\*\*  \* @brief This function is executed in case of error occurrence.  \* @retval None  \*/  **void** **Error\_Handler**(**void**)  {  /\* USER CODE BEGIN Error\_Handler\_Debug \*/  /\* User can add his own implementation to report the HAL error return state \*/  \_\_disable\_irq();  **while** (**1**)  {  }  /\* USER CODE END Error\_Handler\_Debug \*/  }  #ifdef USE\_FULL\_ASSERT  /\*\*  \* @brief Reports the name of the source file and the source line number  \* where the assert\_param error has occurred.  \* @param file: pointer to the source file name  \* @param line: assert\_param error line source number  \* @retval None  \*/  **void** **assert\_failed**(**uint8\_t** \*file, **uint32\_t** line)  {  /\* USER CODE BEGIN 6 \*/  /\* User can add his own implementation to report the file name and line number,  ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/  /\* USER CODE END 6 \*/  }  #endif /\* USE\_FULL\_ASSERT \*/ |