**PEWAKTUAN DAN COUNTING PADA SIMULASI LAP BALAP MOBIL BERBASIS ARM**

****

**Oleh :**

**Hendry Hansel Kurniawan**

**1110141019**

**4 D4 Teknik Elektronika A**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

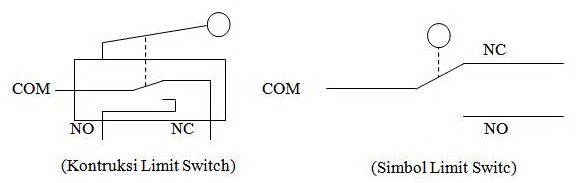
**2017**

1. Tujuan
2. Membuat sebuah mekanisme timeout
3. Mampu mengirim data melalui UART
4. Mampu menggunakan timer untuk mekanisme pewaktuan
5. Landasan Teori
6. Mikrokontroller ARM

Arsitektur **ARM** merupakan arsitektur prosesor 32-bit RISC yang dikembangkan oleh *ARM Limited*. Dikenal sebagai *Advanced RISC Machine* di mana sebelumnya dikenal sebagai *Acorn RISC Machine*. Pada awalnya merupakan *prosesor desktop* yang sekarang didominasi oleh keluarga x86. Namun desain yang sederhana membuat prosesor ARM cocok untuk aplikasi berdaya rendah. Hal ini membuat prosesor ARM mendominasi pasar *mobile electronic* dan *embedded system* di mana membutuhkan daya dan harga yang rendah.

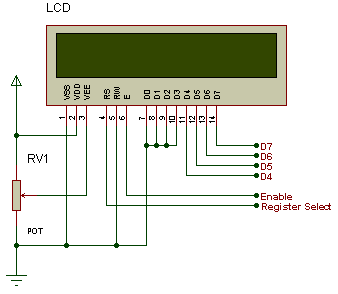
Pada tahun 2007, sekitar 98% dari satu miliar *mobile phone* yang terjual menggunakan setidaknya satu buah prosesor ARM. Dan pada tahun 2009, prosesor ARM mendominasi sekitar 90% dari keseluruhan pasar prosesor 32-bit RISC. Prosesor ARM digunakan di berbagai bidang seperti elektronik umum, termasuk PDA, mobile phone, media player, music player, game console genggam, kalkulator dan periperal komputer seperti hard disk drive dan router.

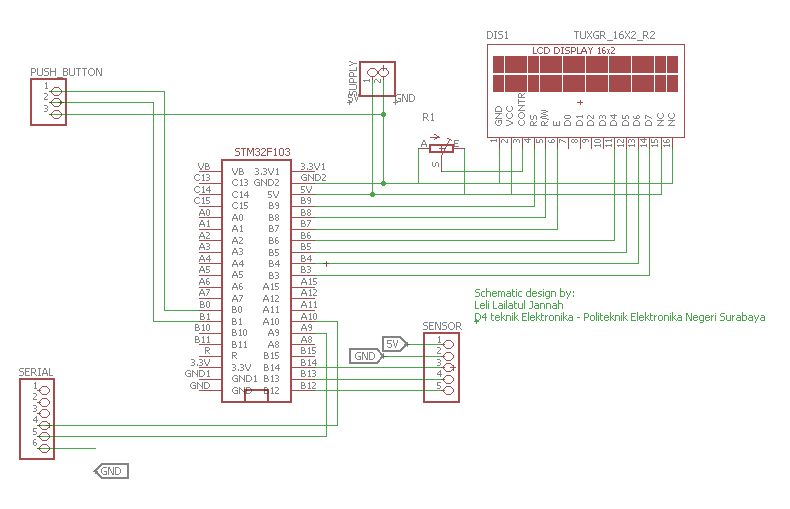
1. Limit Switch



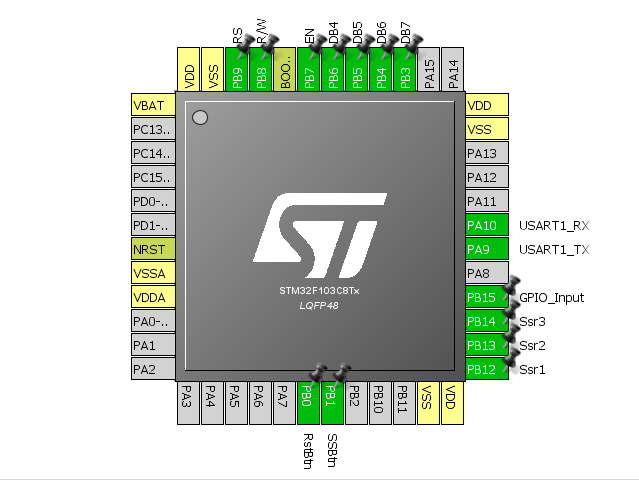
1. Konfigurasi LCD 16X2

Display elektronik adalah salah satu komponen elektronika yang berfungsi sebagai tampilan suatu data, baik karakter, huruf ataupun grafik. LCD (Liquid Cristal Display) adalah salah satu jenis display elektronik yang dibuat dengan teknologi CMOS logic yang bekerja dengan tidak menghasilkan cahaya tetapi memantulkan cahaya yang ada di sekelilingnya terhadap front-lit atau mentransmisikan cahaya dari back-lit. LCD (Liquid Cristal Display) berfungsi sebagai penampil data baik dalam bentuk karakter, huruf, angka ataupun grafik.

1. 
2. Desain Hardware



Desain rangkaian (schematic)



Gambar konfigurasi pin STM32CubeMX

1. Desain Program

* main.c

|  |
| --- |
| /\*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* File Name : main.c  \* Description : Main program body  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*/  /\* Includes ------------------------------------------------------------------\*/  **#include** "main.h"  **#include** "stm32f1xx\_hal.h"  /\* USER CODE BEGIN Includes \*/  **#include** "lcd.h"  **#include**"sensor.h"  /\* USER CODE END Includes \*/  /\* Private variables ---------------------------------------------------------\*/  TIM\_HandleTypeDef htim1;  UART\_HandleTypeDef huart1;  /\* USER CODE BEGIN PV \*/  /\* Private variables ---------------------------------------------------------\*/  u\_int16\_t \_md = 0, \_d = 0, \_m = 0;  u\_int8\_t lap[3] = { 0, 0, 0 };  u\_int8\_t \_status;  u\_int16\_t \_tO[3] = { 0, 0, 0 };  **char** krm[16];  **char** kata[30];  **char** lcd[16];  uint8\_t \_enable = 0;  /\* USER CODE END PV \*/  /\* Private function prototypes -----------------------------------------------\*/  **void** **SystemClock\_Config**(**void**);  **void** **Error\_Handler**(**void**);  **static** **void** **MX\_GPIO\_Init**(**void**);  **static** **void** **MX\_USART1\_UART\_Init**(**void**);  **static** **void** **MX\_TIM1\_Init**(**void**);  /\* USER CODE BEGIN PFP \*/  /\* Private function prototypes -----------------------------------------------\*/  /\* USER CODE END PFP \*/  /\* USER CODE BEGIN 0 \*/  **void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim1) {  **if** (htim1->Instance = TIM1) {  **if** (\_md == 99) {  \_md = 0;  **if** (\_d == 59) {  \_m++;  \_d = 0;  } **else** {  \_d++;  }  } **else** {  \_md++;  }  }  }  /\* USER CODE END 0 \*/  **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();  /\* Configure the system clock \*/  SystemClock\_Config();  /\* Initialize all configured peripherals \*/  MX\_GPIO\_Init();  MX\_USART1\_UART\_Init();  MX\_TIM1\_Init();  /\* USER CODE BEGIN 2 \*/  HAL\_TIM\_Base\_Start\_IT(&htim1);  lcd\_init();  lcd\_putstr("Lap Counter");  /\* USER CODE END 2 \*/  /\* Infinite loop \*/  /\* USER CODE BEGIN WHILE \*/  **while** (1) {  /\* USER CODE END WHILE \*/  /\* USER CODE BEGIN 3 \*/  //HAL\_GPIO\_TogglePin(GPIOC, GPIO\_PIN\_13);  //HAL\_Delay(100);  //lcd\_gotoxy(0, 0);  // lcd\_putstr("Tmbl SS");  **if** (SSBtn()) {  lcd\_clear();  lcd\_gotoxy(0, 0);  **if** (\_enable == 0) {  \_enable = 1;  HAL\_TIM\_Base\_Start\_IT(&htim1);  lcd\_putstr("Started");  } **else** {  \_enable = 0;  HAL\_TIM\_Base\_Stop\_IT(&htim1);  lcd\_putstr("Stoped");  }  }  **if** (RstBtn()) {  \_enable = 0;  HAL\_TIM\_Base\_Stop\_IT(&htim1);  \_md = 0;  \_d = 0;  \_m = 0;  lap[0] = 0;  lap[1] = 0;  lap[2] = 0;  lcd\_clear();  lcd\_gotoxy(0, 0);  lcd\_putstr("Reseted");  }  **if** (\_enable) {  **if** (Ssr1() == 1 || \_tO[0] >= 500) {  **if** (\_tO[0] < 500) {  lap[0]++;  lcd\_gotoxy(0, 0);  **sprintf**(lcd, "M1 l%2d %2d:%2d:%2d", lap[0], \_m, \_d, \_md);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) lcd, 15, 10);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) "\r\n", 2, 10);  lcd\_putstr(lcd);  \_tO[0] = 0;  } **else** {  \_tO[0] = 0;  lcd\_clear();  **sprintf**(krm, "Sensor 1 error");  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) krm, 14, 10);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) "\r\n", 2, 10);  lcd\_gotoxy(0, 0);  lcd\_putstr(krm);  lcd\_gotoxy(0, 1);  lcd\_putstr("Silahkan cek");  }  }  **if** (Ssr2() || \_tO[1] >= 500) {  **if** (\_tO[1] < 500) {  lap[1]++;  lcd\_gotoxy(0, 1);  **sprintf**(lcd, "M2 l%2d %2d:%2d:%2d", lap[1], \_m, \_d, \_md);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) lcd, 15, 10);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) "\r\n", 2, 10);  lcd\_putstr(lcd);  \_tO[1] = 0;  } **else** {  lcd\_clear();  \_tO[1] = 0;  lcd\_clear();  **sprintf**(krm, "Sensor 2 error");  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) krm, 14, 10);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) "\r\n", 2, 10);  lcd\_gotoxy(0, 0);  lcd\_putstr(krm);  lcd\_gotoxy(0, 1);  lcd\_putstr("Silahkan cek");  }  }  **if** (Ssr3() || \_tO[2] >= 500) {  **if** (\_tO[2] < 500) {  lap[2]++;  lcd\_gotoxy(0, 0);  **sprintf**(lcd, "M3 l%2d %2d:%2d:%2d", lap[2], \_m, \_d, \_md);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) lcd, 15, 10);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) "\r\n", 2, 10);  lcd\_putstr(lcd);  \_tO[2] = 0;  } **else** {  lcd\_clear();  \_tO[2] = 0;  **sprintf**(krm, "Sensor 3 error");  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) krm, 14, 10);  HAL\_UART\_Transmit(&huart1, (uint8\_t \*) "\r\n", 2, 10);  lcd\_gotoxy(0, 0);  lcd\_putstr(krm);  lcd\_gotoxy(0, 1);  lcd\_putstr("Silahkan cek");  }  }  }  }  /\* USER CODE END 3 \*/  }  /\*\* System Clock Configuration  \*/  **void** **SystemClock\_Config**(**void**)  {  RCC\_OscInitTypeDef RCC\_OscInitStruct;  RCC\_ClkInitTypeDef RCC\_ClkInitStruct;  /\*\*Initializes the CPU, AHB and APB busses clocks  \*/  RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;  RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;  RCC\_OscInitStruct.HSICalibrationValue = 16;  RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;  RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSI\_DIV2;  RCC\_OscInitStruct.PLL.PLLMUL = RCC\_PLL\_MUL16;  **if** (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != *HAL\_OK*)  {  Error\_Handler();  }  /\*\*Initializes the CPU, AHB and APB busses 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();  }  /\*\*Configure the Systick interrupt time  \*/  HAL\_SYSTICK\_Config(HAL\_RCC\_GetHCLKFreq()/1000);  /\*\*Configure the Systick  \*/  HAL\_SYSTICK\_CLKSourceConfig(SYSTICK\_CLKSOURCE\_HCLK);  /\* SysTick\_IRQn interrupt configuration \*/  HAL\_NVIC\_SetPriority(*SysTick\_IRQn*, 0, 0);  }  /\* TIM1 init function \*/  **static** **void** **MX\_TIM1\_Init**(**void**)  {  TIM\_ClockConfigTypeDef sClockSourceConfig;  TIM\_MasterConfigTypeDef sMasterConfig;  htim1.Instance = TIM1;  htim1.Init.Prescaler = 32000-1;  htim1.Init.CounterMode = TIM\_COUNTERMODE\_UP;  htim1.Init.Period = 20-1;  htim1.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;  htim1.Init.RepetitionCounter = 0;  **if** (HAL\_TIM\_Base\_Init(&htim1) != *HAL\_OK*)  {  Error\_Handler();  }  sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;  **if** (HAL\_TIM\_ConfigClockSource(&htim1, &sClockSourceConfig) != *HAL\_OK*)  {  Error\_Handler();  }  sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;  sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;  **if** (HAL\_TIMEx\_MasterConfigSynchronization(&htim1, &sMasterConfig) != *HAL\_OK*)  {  Error\_Handler();  }  }  /\* USART1 init function \*/  **static** **void** **MX\_USART1\_UART\_Init**(**void**)  {  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  huart1.Init.WordLength = UART\_WORDLENGTH\_8B;  huart1.Init.StopBits = UART\_STOPBITS\_1;  huart1.Init.Parity = UART\_PARITY\_NONE;  huart1.Init.Mode = UART\_MODE\_TX\_RX;  huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;  **if** (HAL\_UART\_Init(&huart1) != *HAL\_OK*)  {  Error\_Handler();  }  }  /\*\* Configure pins as  \* Analog  \* Input  \* Output  \* EVENT\_OUT  \* EXTI  \*/  **static** **void** **MX\_GPIO\_Init**(**void**)  {  GPIO\_InitTypeDef GPIO\_InitStruct;  /\* GPIO Ports Clock Enable \*/  \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();  \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();  /\*Configure GPIO pin Output Level \*/  HAL\_GPIO\_WritePin(GPIOB, DB7\_Pin|DB6\_Pin|DB5\_Pin|DB4\_Pin  |EN\_Pin|R\_W\_Pin|RS\_Pin, *GPIO\_PIN\_RESET*);  /\*Configure GPIO pins : RstBtn\_Pin SSBtn\_Pin Ssr1\_Pin Ssr2\_Pin  Ssr3\_Pin PB15 \*/  GPIO\_InitStruct.Pin = RstBtn\_Pin|SSBtn\_Pin|Ssr1\_Pin|Ssr2\_Pin  |Ssr3\_Pin|GPIO\_PIN\_15;  GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;  GPIO\_InitStruct.Pull = GPIO\_PULLUP;  HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  /\*Configure GPIO pins : DB7\_Pin DB6\_Pin DB5\_Pin DB4\_Pin  EN\_Pin R\_W\_Pin RS\_Pin \*/  GPIO\_InitStruct.Pin = DB7\_Pin|DB6\_Pin|DB5\_Pin|DB4\_Pin  |EN\_Pin|R\_W\_Pin|RS\_Pin;  GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;  GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;  HAL\_GPIO\_Init(GPIOB, &GPIO\_InitStruct);  }  /\* USER CODE BEGIN 4 \*/  /\* USER CODE END 4 \*/  /\*\*  \* @brief This function is executed in case of error occurrence.  \* @param None  \* @retval None  \*/  **void** **Error\_Handler**(**void**)  {  /\* USER CODE BEGIN Error\_Handler \*/  /\* User can add his own implementation to report the HAL error return state \*/  **while** (1) {  }  /\* USER CODE END Error\_Handler \*/  }  **#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**  /\*\*  \* @}  \*/  /\*\*  \* @}  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/ |

* lcd.c

|  |
| --- |
| /\*  \* lcd.c  \*  \* Created on: 18 Nov 2017  \* Author: choirul anam  \*/  #include"lcd.h"  #include "main.h"  #include "stm32f1xx\_hal.h"  void enable()  {  HAL\_GPIO\_WritePin(GPIOB,EN\_Pin,GPIO\_PIN\_SET);//set en  HAL\_Delay(3); // minimal 2ms don't follow databook ;((450ns)<--  HAL\_GPIO\_WritePin(GPIOB,EN\_Pin,GPIO\_PIN\_RESET);//clear en  }  void PORT(uint16\_t count){  //Counter Show  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin,count & (1<<0));  HAL\_GPIO\_WritePin(GPIOB,RW\_Pin,count & (1<<1));  HAL\_GPIO\_WritePin(GPIOB,EN\_Pin,count & (1<<2));  HAL\_GPIO\_WritePin(GPIOB,D4\_Pin,count & (1<<4));  HAL\_GPIO\_WritePin(GPIOB,D5\_Pin,count & (1<<5));  HAL\_GPIO\_WritePin(GPIOB,D6\_Pin,count & (1<<6));  HAL\_GPIO\_WritePin(GPIOB,D7\_Pin,count & (1<<7));  }  void lcd\_init()  {  HAL\_GPIO\_WritePin(GPIOB,EN\_Pin,GPIO\_PIN\_RESET);  HAL\_Delay(20);  //step1  PORT(0x30);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  HAL\_Delay(5);  // step2  PORT(0x30);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  HAL\_Delay(1);  // step3  PORT(0x30);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  //next init  PORT(0x20);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  PORT(0x20);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  PORT(0x80);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  PORT(0x00);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();// display on  PORT(0xE0);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  PORT(0x00);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable(); // entry mode address inc and cursor shift right  PORT(0x60);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  PORT(0x00);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable(); // display clear  PORT(0x10);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable(); // display clear  }  void lcd\_clear(void)  {  PORT(0x00);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable(); // display clear  PORT(0x10);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable(); //  }  void lcd\_gotoxy(char j,char i)  {  unsigned char posisi,kar2;  posisi=(i\*0x40)+j;  //posisi=(i>0?0x40+j:j);  kar2= posisi & 0xF0;//0xF0  PORT(kar2);  HAL\_GPIO\_WritePin(GPIOB,D7\_Pin,GPIO\_PIN\_SET);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  posisi= (posisi<<4)|(posisi>>4); // swap temp  kar2= posisi & 0xF0;//0xF0  PORT(kar2);  HAL\_GPIO\_WritePin(GPIOB,RS\_Pin|RW\_Pin,GPIO\_PIN\_RESET);enable();  }  void lcd\_putchar(unsigned char kar)  {  unsigned char kar2;  //kar2=(kar|(PORT\_LCD&0x0F))& (PORT\_LCD|0xF0); //save control in bit 0-3  //kar2= (kar & 0xF0) + (PORT\_LCD &0x0F);  //CLRBIT(kar2,EN);  kar2=kar & 0xF0;  PORT(kar2);HAL\_GPIO\_WritePin(GPIOB,RS\_Pin,1);HAL\_GPIO\_WritePin(GPIOB,RW\_Pin,0);enable(); //  kar=(kar<<4|kar>>4);  kar2 = kar & 0xF0;  PORT(kar2);HAL\_GPIO\_WritePin(GPIOB,RS\_Pin,1);HAL\_GPIO\_WritePin(GPIOB,RW\_Pin,0);enable(); // display clear  }  void lcd\_putstr(char \*s)  {  char i=0;  char c;  while((c=\*(s+(i++)))!=0)  lcd\_putchar(c);  } |

* sensor.c

|  |
| --- |
| /\*  \* sensor.c  \*  \* Created on: 9 Nov 2017  \* Author: tinova  \*/  #include "sensor.h"  uint8\_t SSBtn(void)  {  uint8\_t FlagDetect=0;  if(HAL\_GPIO\_ReadPin(SSBtn\_GPIO\_Port,SSBtn\_Pin)== GPIO\_PIN\_RESET) {  HAL\_Delay(20);  if(HAL\_GPIO\_ReadPin(SSBtn\_GPIO\_Port,SSBtn\_Pin)== GPIO\_PIN\_RESET) {  while(HAL\_GPIO\_ReadPin(SSBtn\_GPIO\_Port,SSBtn\_Pin)== GPIO\_PIN\_RESET);  FlagDetect=1;  }  }  return FlagDetect;  }  uint8\_t RstBtn(void)  {  uint8\_t FlagDetect=0;  if(HAL\_GPIO\_ReadPin(RstBtn\_GPIO\_Port,RstBtn\_Pin)== GPIO\_PIN\_RESET) {  HAL\_Delay(20);  if(HAL\_GPIO\_ReadPin(RstBtn\_GPIO\_Port,RstBtn\_Pin)== GPIO\_PIN\_RESET) {  while(HAL\_GPIO\_ReadPin(RstBtn\_GPIO\_Port,RstBtn\_Pin)== GPIO\_PIN\_RESET);  FlagDetect=1;  }  }  return FlagDetect;  }  uint8\_t Ssr1(void)  {  uint8\_t FlagDetect=0;  if(HAL\_GPIO\_ReadPin(Ssr1\_GPIO\_Port,Ssr1\_Pin)== GPIO\_PIN\_RESET) {  HAL\_Delay(20);  if(HAL\_GPIO\_ReadPin(Ssr1\_GPIO\_Port,Ssr1\_Pin)== GPIO\_PIN\_RESET) {  while(HAL\_GPIO\_ReadPin(Ssr1\_GPIO\_Port,Ssr1\_Pin)== GPIO\_PIN\_RESET);  FlagDetect=1;  }  }  return FlagDetect;  }  uint8\_t Ssr2(void)  {  uint8\_t FlagDetect=0;  if(HAL\_GPIO\_ReadPin(Ssr2\_GPIO\_Port,Ssr2\_Pin)== GPIO\_PIN\_RESET) {  HAL\_Delay(20);  if(HAL\_GPIO\_ReadPin(Ssr2\_GPIO\_Port,Ssr2\_Pin)== GPIO\_PIN\_RESET) {  while(HAL\_GPIO\_ReadPin(Ssr2\_GPIO\_Port,Ssr2\_Pin)== GPIO\_PIN\_RESET);  FlagDetect=1;  }  }  return FlagDetect;  }  uint8\_t Ssr3(void)  {  uint8\_t FlagDetect=0;  if(HAL\_GPIO\_ReadPin(Ssr3\_GPIO\_Port,Ssr3\_Pin)== GPIO\_PIN\_RESET) {  HAL\_Delay(20);  if(HAL\_GPIO\_ReadPin(Ssr3\_GPIO\_Port,Ssr3\_Pin)== GPIO\_PIN\_RESET) {  while(HAL\_GPIO\_ReadPin(Ssr3\_GPIO\_Port,Ssr3\_Pin)== GPIO\_PIN\_RESET);  FlagDetect=1;  }  }  return FlagDetect;  } |

1. Hasil Percobaan

Gambar run awal

Gambar ketika dilakukan reset

Ketika dilakukan start/stop

1. Analisa

Pada project ulangan tengah semester ini, dilakukan pembuatan sebuah program untuk proses time counting pada racing lap (balapan mobil). Menggunakan 3 buah track / jalur balapan dan digunakan Tamiya sebagai simulasi mobil.

Dalam project ini, digunakan arm stm32f103C8TX dengan menggunakan 3 buah sensor limitswitch yang dipasang pada tiap-tiap track nya. Tujuannya, ketika mobil melewati track akan menyentuh liit switch sehingga counting dilakukan. Terdapat pula 2 buah push button untuk start/stop dan reset timer dan perhitungan lab digunakan lcd sebagai tambahan tampilan dalam project ini.

Pada project ini, hal pertama yang dilakukan adalah membuat schematic dari rangkaian. Kemudian setelah selesai dilakukan pembuatan rangkaian, dilakukan pembuatan hardware serta pembuatan casing / kotak kemasan. Setrelah itu, dilakukan pengecekan pada setiap komponen yang telah disolder, salah satunya dengan melakukan pengecekan pada data yang dikeluarkan oleh limitswitch. Kemudian dilakukan pengecekan pada push button, apakah sudah bekerja dengan baik dan pengecekan pada kaki-kaki header apakah sudah tersambung dengan benar apa tidak. Setelah hardware telah terpenuhi maka dilakukan proses pemrograman yang dilakukan dengan menggunakan program seperti pada program diatas.

Proses pemrograman dilakukan dengan inisialisasi pin-pin dan port – port yang akan digunakan dengan menggunakan STMcubeMX. Kemudian, setelah dilakukan generate, maka otomatid eclipse aan terbuka dengan konfigurasi pin-pin yang telah diatur tadi.

Proses selanjutnya digunakan program timer untuk melakukan pembuatan pewaktuan dalam proses pertandingan mobil. Dalam proses timer ini digunakan prescaller yang digunakan untuk mendapatkan waktu per ticknya, sehingga dapat digunakan dalam proses pewaktuan seperti pada gambar dibawah ini :

|  |
| --- |
| **void** **HAL\_TIM\_PeriodElapsedCallback**(TIM\_HandleTypeDef \*htim2) {  **if** (htim2->Instance = TIM2) {  **if** (\_md == 99) {  \_md = 0;  **if** (\_d == 59) {  \_m++;  \_d = 0;  } **else** {  \_d++;  }  } **else** {  \_md++;  }  }  }  /\* USER CODE |

Kemudian dilakukan pembuatan uart dengan uart1 untuk melakukan komunikasi serial kepada komputer dengan menggunakan baudrate sebesar 115200 agar sesuai dengan baudrate dari komputer 115200. Uart dilakukan untuk mengirimkan data pewaktuan, lap yang dilalui mobil serta pesan error apabila terjadi error pada sensor yang selanjutnya ditampilkan pada serial monitor pada komputer. Berikut konfigurasi UART:

|  |
| --- |
| **static** **void** **MX\_USART1\_UART\_Init**(**void**)  {  huart1.Instance = USART1;  huart1.Init.BaudRate = 115200;  huart1.Init.WordLength = UART\_WORDLENGTH\_8B;  huart1.Init.StopBits = UART\_STOPBITS\_1;  huart1.Init.Parity = UART\_PARITY\_NONE;  huart1.Init.Mode = UART\_MODE\_TX\_RX;  huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;  huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;  **if** (HAL\_UART\_Init(&huart1) != *HAL\_OK*)  {  Error\_Handler();  }  } |

Setelah itu, dilakukan pembacaan sensor yang dilakukan dengan menggunakan mekanisme timeout. Mekanisme timeout yang dilakukan diperlukan untuk mengetahui adanya hardware yang error dengan mekanismenya yaitu pertama menentukan inisialisasi timeout dan berapa lamanya waktu yang dibutuhkaan saat akan timeout, kemudian timeout dimasukkan kedalam program saat sesudah program penantian bouncing selama 20 detik. Saat sensor mendeteksi tidak ada bouncing dan dilakukan pengecekan kembali sensor masih mendeteksi object maka digunggu sampai object meninggalkan sensor, apabila object telah lewat dan masih selama waktu belum timeout maka akan didefinisikan ada object yang lewat. Namun apabila object terdeteksi dan sudah melewati masa time out maka, serial akan menuliskan atau mengirimkan data berupa sensor error dan tidak akan melakukan counting karena, program akan kembali menuju flag = 0 sama seperti mekanisme sensor, tombol start/stop dan reset juga dilakuakan mekanisme yang sama, kemudian pada setiap sensor dilakukan pembuatan mekanisme bouncing pada setiap sensor dan tombol yang digunakan untuk membuat sensor tidak mengcounting berkali-kali pada setiap sensing dan penekanan tombol. Untuk melakukan penampilan pada lcd digunakan program seperti pada program diatas, program lcd digunakan dengan pemanggilan sub routin dari lcd, dan pada setiap data yang akan ditampilkan dilakukan pemanggilan sub routin lcd\_string untuk menampilkan data string. Mekanisme ini memerlukan penyimpanan data pada array untuk mempermudah penampilan data. Namun, hasil pada projek kali ini belum mencapai maksimal diduga karena kesalahan pada hardware yang dilakukan.

1. Kesimpulan

Dari project sistem embedded yang telah dilakukan, dapat ditarik kesimpulan bahwa:

* Timer pada arm dapat digunakan sebagai pewaktuan yang sebenarnya pada simulasi racing lap dengan menggunkan sumber clock internal
* Pad sistem, diberi mekanisme anti bouncing dilakukan untuk menghindari bouncing yang terjadi pada setiap sensor dan push button
* Selain itu diberi mekanisme timeout sehingga ketika terjadi kesalahan pada sensor, maka dapat diketahui. Sehinga, kita dapat mengetahui sensor mana yang tidak berfungsi
* Program juga dibuat berstruktur, sehingga lebih mudah dan ringkas untuk dipahami

1. Lampiran Barcode

github



archive

