ภาคผนวก ก

โปรแกรมไมโครคอนโทรลเลอร์ STM32

โปรแกรมจานรักษาสมดุล

```
#include "stm32f10x_lib.h"        // เพิ่มไลบรารี่สำหรับบอร์ด STM32
#include "RCC_setup.h"            // เพิ่มไลบรารี่ RCC
#include "NVIC_setup.h"           // เพิ่มไลบรารี่ NVIC
#include "GPIO_setup.h"           // เพิ่มไลบรารี่ GPIO
#include "UART.h"                 // เพิ่มไลบรารี่ UART
#include "ADC.h"                  // เพิ่มไลบรารี่ ADC
#include "PWM.h"                  // เพิ่มไลบรารี่ UART
#include "Timer.h"                // เพิ่มไลบรารี่ Timer
#include "FuzzyPID.h"             // เพิ่มไลบรารี่ FuzzyPID
#include "RCServo.h"              // เพิ่มไลบรารี่ RC Servo
#include "Ultrasonics.h"          // เพิ่มไลบรารี่ Ultrasonics
#include "I2C_HMC6352.h"          // เพิ่มไลบรารี่ I2C_HMC6352 h
#include <stdio.h>                // เพิ่มไลบรารี่ stdio n
#include <stdlib.h>               // เพิ่มไลบรารี่ stdlib
// กำหนดค่าตัวแปรสำหรับการใช้งานทั่วไป
volatile unsigned int ms = 0;
volatile int n = 0;
volatile int setpointX = 1556;
volatile int setpointY = 1870;
bool flag = 0;
// กำหนดค่าตัวแปรสำหรับอ่านค่าจากคอมพอร์ต
char Rx1_Read;
// ฟังก์ชั่นในการแสดงค่าข้อมูลทางคอมพอร์ต
void PrintData(void){
        if(flag ==0)
          {
                printf("@D,%.1f,%d,%d,%d,E\r\n",Compass

              ,ultrasonicLeft(ADC_RegularConvertedValueTab[0])
```

```
                    ,ultrasonicFront(ADC_RegularConvertedValueTab[1])

                    ,ultrasonicRight(ADC_RegularConvertedValueTab[2]));
                }
              if(flag ==1)
                {
    printf("$P,%.2f,%.2f,%.2f,%.2f,%.2f,%1.3f,%1.3f,%1.3f,%1.3f,%1.3f,%1.3f,E\n"

,PID_1,PID_2,roll,pitch,Yaw,KP_1,KI_1,KD_1,KP_2,KI_2,KD_2);
                }
    }
    // ฟังก์ชั่นอินเตอร์รัพท์ของไทเมอร์ 1
    void TIM1_UP_IRQHandler(void)
    {
        if(ms>=2)     // อัพเดทค่าทุกๆ 2 มิลลิวินาที
        {
          ms=0;
          PID(setpointX,setpointY);
        }
        ms++;
       // เคลียร์ค่าบิตอินเตอร์รัพท์ไทเมอร์ 1
        TIM_ClearITPendingBit(TIM1,TIM_IT_Update);
    }
    // ฟังก์ชั่นอินเตอร์รัพท์ของไทเมอร์ 2
    void TIM2_IRQHandler(void)
    {
       // แสดงค่าข้อมูลทุกๆ 50 มิลลิวินาที
       if(n>=5)
          {
```

```
            PrintData();
          }
        n++;
         // เคลียร์ค่าบิตอินเตอร์รัพท์ไทเมอร์ 2
        TIM_ClearITPendingBit(TIM2,TIM_IT_Update);
}
// ฟังก์ชั่นหลักของโปรแกรม
int main()
{

   RCC_setup();                          // กำหนดฟังก์ชั่นของ RCC
   NVIC_setup();                         // กำหนดฟังก์ชั่นของ NVIC
   GPIO_setup();                         // กำหนดฟังก์ชั่นของ GPIO
   RCMotor_setup();                      // กำหนดฟังก์ชั่นของ RC Motor
   USART1_setup(115200);                 // กำหนดฟังก์ชั่นของ USART1
   USART2_setup(115200);          // กำหนดฟังก์ชั่นของ USART1 สำหรับ IMU เซนเซอร์
   ADC_setup();                          // กำหนดฟังก์ชั่นของ ADC
   //I2C_setup();                        // กำหนดฟังก์ชั่นของ I2C สำหรับเข็มทิศอิเล็กทรอนิกส์
   TIMER_setup();                        // กำหนดฟังก์ชั่นของ TIMER
   TIM_Cmd(TIM1, DISABLE);       // ปิดการใช้งานอินเตอร์รัพท์ไทเมอร์ 1
   TIM_Cmd(TIM2, DISABLE);       // ปิดการใช้งานอินเตอร์รัพท์ไทเมอร์ 2
   printf("Hello ARM Cortex-M3\r\n");  // แสดงข้อความ Hello ARM Cortex-M3
   RC_Motor_Set(1556,1870);              // กำหนดค่าเริ่มต้นให้กับอาร์ซีเซอร์โวมอเตอร์
   // กำหนดค่าเริ่มต้นให้กับสมการพีไอดี
   KP_1=0.2;
   KI_1=15.2;
   KD_1=0.01;
   KP_2=0.1;
   KI_2=25.0;
   KD_2=0.002;
```

```
while(1) // วนลูปไม่รู้จบ
  {
    Rx1_Read=usart1_getc();
    switch(Rx1_Read)
    {
       case 't':
          TIM_Cmd(TIM1, DISABLE);
          TIM_Cmd(TIM2, DISABLE);
          RC_Motor_Set(1556,1870);
          break;
       case 'g':
          TIM_Cmd(TIM1, DISABLE);
          TIM_Cmd(TIM2, DISABLE);
          RC_Motor_Set(1556,1870);
          KP_1 =usart1_Gain();
          KI_1 =usart1_Gain();
          KD_1 =usart1_Gain();
          KP_2 =usart1_Gain();
          KI_2 =usart1_Gain();
          KD_2 =usart1_Gain();
          setpointX =usart1_getnum();
          setpointY =usart1_getnum();
        printf("kp_1:%1.3f ki_1:%1.3f kd_1:%1.3f kp_2:%1.3f
                  ki_2:%1.3fkd_2:%1.3f\r\n"
                  ,KP_1,KI_1,KD_1,KP_2,KI_2,KD_2);
          TIM_Cmd(TIM1, ENABLE);
          TIM_Cmd(TIM2, ENABLE);
          break;
       case 's':
          KP_1=0.2;
```

```
            KI_1=15.2;
            KD_1=0.01;
            KP_2=0.1;
            KI_2=25.0;
            KD_2=0.002;
            setpointX = 1556;
            setpointY = 1870;
            TIM_Cmd(TIM1, ENABLE);
            TIM_Cmd(TIM2, ENABLE);
            break;
          case 'l':
            TIM_Cmd(TIM2, DISABLE);
            RC_Motor_Set(1556,1870);
            Rx1_Read=usart1_getc();
            if(Rx1_Read =='o') { flag = 0;}
            if(Rx1_Read =='z') { flag = 1;}
            TIM_Cmd(TIM2, ENABLE);
            break;
        }
      }
  //return 0;
}
```

โปรแกรมขับมอเตอร์หุ่นยนต์

```
#include "stm32f10x_lib.h"          // เพิ่มไลบรารี่สำหรับบอร์ด STM32
#include "RCC_setup.h"              // เพิ่มไลบรารี่ RCC_setup
#include "NVIC_setup.h"             // เพิ่มไลบรารี่ NVIC_setup
#include "GPIO_setup.h"             // เพิ่มไลบรารี่ GPIO_setup
#include "Systick.h"                // เพิ่มไลบรารี่ Systick
#include "UART.h"                   // เพิ่มไลบรารี่ UART
#include "PWM.h"                    // เพิ่มไลบรารี่ PWM
#include "Timer.h"                  // เพิ่มไลบรารี่ Timer
#include "ADC.h"                    // เพิ่มไลบรารี่ ADC
#include "DCMotor.h"                // เพิ่มไลบรารี่ DC Motor
#include "Currentsensor.h"          // เพิ่มไลบรารี่ Current sensor
#include "Battery.h"                // เพิ่มไลบรารี่ Battery
#include "PIDs.h"                   // เพิ่มไลบรารี่ PID
#include <stdio.h>                  // เพิ่มไลบรารี่ stdio
#include <stdlib.h>                 // เพิ่มไลบรารี่ stdlib
// กำหนดค่าตัวแปรสำหรับมอเตอร์ด้านขวา
volatile unsigned int time_ms = 0;
volatile unsigned int time = 0;
volatile float Pulse1 = 0.0;
volatile float Pulse_1 = 0.0;
volatile double Position1 = 0.0;
volatile float RPM_M1 = 0.0;
volatile float Feedback_1 = 0;
volatile int RPM_Setpoint1 = 0;
// กำหนดค่าตัวแปรสำหรับมอเตอร์ด้านซ้าย
volatile float Pulse2 = 0.0;
volatile float Pulse_2 = 0.0;
volatile double Position2 = 0.0;
volatile float RPM_M2 = 0.0;
```

```
volatile float Feedback_2 = 0;
volatile int RPM_Setpoint2 = 0;
char Rx1_Read = 0;
// กำหนดค่าตัวแปรสำหรับฟังก์ชั่นพีไอดี
float Kp_1 = 0;
float Ki_1 = 0;
float Kd_1 = 0;
float Kp_2 = 0;
float Ki_2 = 0;
float Kd_2 = 0;
// ฟังก์ชั่นการหน่วงเวลา
void delay(unsigned long ms)
{
  volatile unsigned long i,j;
  for (i = 0; i < ms; i++ )
  for (j = 0; j < 5525; j++ );
}
// ฟังก์ชั่นการแสดงค่าข้อมูลผ่านทางคอมพอร์ต
void PrintData(void)
{
                printf("$D,%.1f,%.1f,%.1f,%.1f,%.1f,%.1f,E\r\n"
                                ,Position1,Feedback_1,Position2,Feedback_2
                                ,GetCurrentM1(ADC_RegularConvertedValueTab[4])
                                ,GetCurrentM2(ADC_RegularConvertedValueTab[5]));

}
```

```
// ฟังกั่นการแสดงค่าข้อมูลแบตเตอรี่
void PrintBattery(void)
{
    int i;
    for(i = 0;i<10;i++)
     {
  printf("@T,%.1f,%.1f,%.1f,%.1f,E\r\n"
            ,Read_Battery1(ADC_RegularConvertedValueTab[0])

            ,Read_Battery2(ADC_RegularConvertedValueTab[1])

            ,Read_Battery3(ADC_RegularConvertedValueTab[2])

            ,Read_Battery4(ADC_RegularConvertedValueTab[3]));
     }

}
// ฟังก์ชั่นอินเตอร์รัพท์ของไทเมอร์ 1
void TIM1_UP_IRQHandler(void)    // อัพเดทค่าทุกๆ 1 มิลลิวินาที
{
        Pulse1  = TIM_GetCounter(TIM2)-32768;
        TIM_SetCounter(TIM2,32768);
        Position1 += Pulse1;
        if(Pulse1 < 0 )
        {
          RPM_M1    = RPM_M1 +(-1*Pulse1);
        }
        else
        RPM_M1  += Pulse1;
        Pulse_1 = Pulse1;
```

```
   Pulse2  = TIM_GetCounter(TIM3)-32768;
   TIM_SetCounter(TIM3,32768);
   Position2 += Pulse2;
   if(Pulse2 < 0 )
   {
     RPM_M2    = RPM_M2 +(-1*Pulse2);
   }
   else
   RPM_M2  += Pulse2;
   Pulse_2 = Pulse2;
if(time_ms>=10)
{
   time_ms = 0;
   // คำนวณค่ารอบต่อนาทีของมอเตอร์ด้านซ้าย
   RPM_M1 = RPM_M1/1600;              // หารค่าพัลต์เอ็นโค้ดเดอร์เพื่อให้ได้ 1 พัลต์
   RPM_M1 =(RPM_M1*6000);
   Feedback_1 = RPM_M1;
   RPM_M1 =0;
   // คำนวณค่ารอบต่อนาทีของมอเตอร์ด้านขวา
   RPM_M2 = RPM_M2/1600;              // หารค่าพัลต์เอ็นโค้ดเดอร์เพื่อให้ได้ 1 พัลต์
   RPM_M2 =(RPM_M2*6000);
   if(RPM_M1 < 0 ){-RPM_M1;}
   Feedback_2 = RPM_M2;
   RPM_M2 =0;
 }
```

```
      // เรียกใช้งานฟังก์พีไอดี
      PID(RPM_Setpoint1,RPM_Setpoint2,Feedback_1,Feedback_2);
      time_ms++;
      TIM_ClearITPendingBit(TIM1,TIM_IT_Update);
}
// เคลียร์ค่าข้อมูลเอ็นโค้ดเดอร์
void ClearEncoder()
{
       Pulse1=0;
       Pulse2=0;
       Position1=0;
       Position2=0;
}
// ฟังก์ชั่นรีเซตค่าวอตช์ด็อก
void WatchdogInit(void)
{
    if (RCC->CSR & (1<<29)) {
    RCC->CSR |= (1<<24);
    printf("\r\n Clera WatchDog \n");
  }
}
void WatchDogRefresh(void)
{
    IWDG->KR  = 0xAAAA;                    // รีโหลดค่าวอตช์ด็อก
}
```

```
// ฟังก์ชั่นซิสเต็มติกไทเมอร์
void SysTickHandler(void)
{
        // แสดงค่าข้อมูลออกทางคอมพอร์ต
         PrintData();
         // เคลียร์ค่าวอตช์ด็อก
         WatchDogRefresh();
}
// ฟังก์ชั่นหลักของโปรแกรม
int main()
{
   RCC_setup();                    // กำหนดฟังก์ชั่นของ RCC
   NVIC_setup();                   // กำหนดฟังก์ชั่นของ NVIC
   GPIO_setup();                   // กำหนดฟังก์ชั่นของ GPIO
   DCMotor_setup();                // กำหนดฟังก์ชั่นของ DC Motor
   USART1_setup(115200);           // กำหนดฟังก์ชั่นของ USART1
   TIMER_setup();                  // กำหนดฟังก์ชั่นของ TIMER
   ADC_setup();                    // กำหนดฟังก์ชั่นของ ADC
   Encoder_setup();                // กำหนดฟังก์ชั่นของ Encoder
   TIM_SetCounter(TIM2,32768);   // กำหนดฟังก์ชั่นของไทเมอร์  2
   TIM_SetCounter(TIM3,32768);   // กำหนดฟังก์ชั่นของไทเมอร์ 3
   TIM_Cmd(TIM1, DISABLE);         // ปิดการใช้งานอินเตอร์รัพท์ไทเมอร์ 1
   ClearEncoder();                 // เคลียร์ค่าข้อมูลเอ็นโค้ดเดอร์
   SYSTICK_setup();                // กำหนดฟังก์ชั่นของซิสเต็มติกไทเมอร์
   WatchdogInit();                 // กำหนดฟังก์ชั่นของวอตช์ด็อก
   // แสดงค่าข้อความออกทางคอมพอร์ต
   printf("\r\n Hello STM32 DC Motor Down \n");
   SysTick_CounterCmd(SysTick_Counter_Disable);
   SetGain(0.1,2.0,0.001,0.1,2.0,0.001);
   SysTick_CounterCmd(SysTick_Counter_Enable);
```

```
while(1)    // วนลูปไม่รู้จบ
  {
    Rx1_Read = usart1_getc();
    switch (Rx1_Read)
          {
      case 'g':
          TIM_Cmd(TIM1, DISABLE );
          RPM_Setpoint1 =usart1_getnum();
          RPM_Setpoint2 =usart1_getnum();
          if(RPM_Setpoint1>150){RPM_Setpoint1 =0;}
          if(RPM_Setpoint2>150){RPM_Setpoint2 =0;}
          // wait for close loop control
          delay(2000);
          Setmotor_Go(RPM_Setpoint1,RPM_Setpoint2);
          TIM_Cmd(TIM1, ENABLE);
          break;
      case 'b':
          TIM_Cmd(TIM1, DISABLE );
          RPM_Setpoint1 =usart1_getnum();
          RPM_Setpoint2 =usart1_getnum();
          if(RPM_Setpoint1>150){RPM_Setpoint1 =0;}
          if(RPM_Setpoint2>150){RPM_Setpoint2 =0;}
          Setmotor_Back(RPM_Setpoint1,RPM_Setpoint2);
          TIM_Cmd(TIM1, ENABLE);

          break;
      case 'r':
          TIM_Cmd(TIM1, DISABLE );
          RPM_Setpoint1 =usart1_getnum();
          RPM_Setpoint2 =usart1_getnum();
```

```
            if(RPM_Setpoint1>150){RPM_Setpoint1 =0;}
            if(RPM_Setpoint2>150){RPM_Setpoint2 =0;}
            Setmotor_Right(RPM_Setpoint1,RPM_Setpoint2);
            Pulse1 =0;
            Pulse2 =0;
            TIM_Cmd(TIM1, ENABLE);
            break;
        case 'l':
            TIM_Cmd(TIM1, DISABLE );
            RPM_Setpoint1 =usart1_getnum();
            RPM_Setpoint2 =usart1_getnum();
            if(RPM_Setpoint1>150){RPM_Setpoint1 =0;}
            if(RPM_Setpoint2>150){RPM_Setpoint2 =0;}
            Setmotor_Left(RPM_Setpoint1,RPM_Setpoint2);
            Pulse1 =0;
            Pulse2 =0;
            TIM_Cmd(TIM1, ENABLE);
            break;
        case 't':
            TIM_Cmd(TIM1, DISABLE );
            RPM_Setpoint1 = 0;
            RPM_Setpoint2 = 0;
            Setmotor_Stop(0,0);
            TIM_Cmd(TIM1, ENABLE);
            break;
        case 'd':
            TIM_Cmd(TIM1, DISABLE);
            SysTick_CounterCmd(SysTick_Counter_Disable);

}
```