PID Control and Artificial Intelligence Codes :

sparton_code.c:

```
/*============================================
============================
  File:    sparton_code.c
  Purpose: Self-balancing robot using PID & AI on Sparton
(ARM Cortex-M)
  Author:  Mahmoud Ali Hassan
  Date:    2015
============================================
============================*/

#include "stm32f4xx_hal.h"
#include "mpu6050.h"
#include "ai_module.h"

/* PID constants */
#define Kp  35.0f
#define Ki   5.0f
```

```
#define Kd  15.0f

/* Target angle (upright) */
static const float target_angle = 0.0f;

/* Handles */
I2C_HandleTypeDef hi2c1;
TIM_HandleTypeDef htim1;

/* PID variables */
float angle, gyro;
float error, prev_error = 0.0f;
float integral = 0.0f;
float derivative;
float control_output;

/* Prototypes */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
```

```
static void MX_TIM1_Init(void);
void set_PWM_Sparton(float pwm);
void mpu6050_read(float* pAngle, float* pGyro);

int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_TIM1_Init();

  /* Start PWM on channel 1 & 2 */
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2);

  /* Main control loop */
  while (1)
  {
    /* 1. Read IMU */
```

```
    mpu6050_read(&angle, &gyro);

    /* 2. PID computation */
    error = target_angle - angle;
    integral += error * 0.01f;              // dt = 10 ms
    derivative = (error - prev_error) / 0.01f;
    control_output = Kp * error
                   + Ki * integral
                   + Kd * derivative;
    prev_error = error;

    /* 3. Apply PWM to motors */
    set_PWM_Sparton(control_output);

    /* 4. AI decision for path planning */
    ai_decision();

    /* 5. Loop delay */
    HAL_Delay(10);
}
```

```
}

/* Stub: configure system clock (CubeMX generated) */
void SystemClock_Config(void) { /* ... */ }

/* Stub: init GPIO pins (CubeMX generated) */
static void MX_GPIO_Init(void) { /* ... */ }

/* Stub: init I2C1 for MPU6050 */
static void MX_I2C1_Init(void)
{
  hi2c1.Instance           = I2C1;
  hi2c1.Init.ClockSpeed    = 100000;
  hi2c1.Init.DutyCycle     = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1   = 0;
  hi2c1.Init.AddressingMode  =
I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode =
I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2   = 0;
```

```
  hi2c1.Init.GeneralCallMode =
I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode   =
I2C_NOSTRETCH_DISABLE;
  HAL_I2C_Init(&hi2c1);
}

/* Stub: init TIM1 for PWM (CubeMX generated) */
static void MX_TIM1_Init(void)
{
  TIM_OC_InitTypeDef sConfigOC = {0};

  htim1.Instance           = TIM1;
  htim1.Init.Prescaler     = 84 - 1;     // 1 MHz timer
clock
  htim1.Init.CounterMode   =
TIM_COUNTERMODE_UP;
  htim1.Init.Period        = 2000 - 1;   // 500 Hz PWM
  htim1.Init.ClockDivision     =
TIM_CLOCKDIVISION_DIV1;
  HAL_TIM_PWM_Init(&htim1);
```

```
  /* Channel 1 configuration */
  sConfigOC.OCMode      = TIM_OCMODE_PWM1;
  sConfigOC.Pulse       = 0;
  sConfigOC.OCPolarity  = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode  = TIM_OCFAST_DISABLE;
  HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1);

  /* Channel 2 duplicate for second motor */
  HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_2);
}

/* Set PWM duty-cycle based on control output */
void set_PWM_Sparton(float pwm)
{
  uint32_t pulse = (uint32_t)fminf(fmaxf(0.0f, pwm), 1999.0f);
  __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, pulse);
  __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_2, pulse);
```

```
}
```

pic_code.c

```
/*==========================================
==============================
  File:    pic_code.c
  Purpose: Self-balancing robot using PID & AI on
PIC18F46K22
  Author:  Mahmoud Ali Hassan
  Date:    2015
==========================================
============================*/

#include <xc.h>
#include "mpu6050.h"
#include "ai_module.h"

/* Configuration bits */
#pragma config FOSC = HS        // High-speed crystal
#pragma config WDTE = OFF       // Watchdog Timer off
#pragma config PWRTE = ON       // Power-up Timer on
#pragma config MCLRE = ON       // MCLR pin enabled
```

```
#pragma config LVP = OFF       // Low-voltage programming off
#pragma config BOREN = ON      // Brown-out reset on
#pragma config CP = OFF        // Code protection off
#pragma config WRT = OFF       // Flash write protection off

/* Oscillator frequency for __delay_ms */
#define _XTAL_FREQ 12000000UL

/* PID constants */
#define Kp  35.0f
#define Ki   5.0f
#define Kd  15.0f

/* Target angle */
static const float target_angle = 0.0f;

/* PID variables */
float angle, error, last_error = 0.0f;
float integral = 0.0f, derivative;
```

```
/* Function prototypes */
void init_system(void);
void init_I2C(void);
void init_PWM(void);
void set_motor_pwm(unsigned int val);

void main(void)
{
   init_system();

   while (1)
   {
      /* 1. Read MPU6050 */
      mpu6050_read(&angle, NULL);

      /* 2. PID */
      error = target_angle - angle;
      integral   += error;
      derivative  = error - last_error;
```

```
        last_error  = error;

        unsigned int pwm = (unsigned int)(Kp*error +
Ki*integral + Kd*derivative);
        set_motor_pwm(pwm);

        /* 3. AI decision */
        AI_Process();

        /* 4. Delay 10 ms */
        __delay_ms(10);
    }
}

/* Initialize I2C for MPU6050 */
void init_I2C(void)
{
    SSPCON1 = 0x28;  // I2C Master mode
    SSPADD  = ((_XTAL_FREQ/100000UL)/4) - 1;
    SSPSTAT = 0x00;
```

```
}

/* Initialize PWM1 */
void init_PWM(void)
{
   /* Configure CCP1 and CCP2 modules for PWM */
   TRISC2 = 0;    // CCP1
   TRISC1 = 0;    // CCP2
   PR2    = 199;  // PWM period
   T2CON  = 0x04; // Timer2 on, prescaler = 1
   CCP1CON = 0x0C;
   CCP2CON = 0x0C;
}

/* Set both motors to same PWM */
void set_motor_pwm(unsigned int val)
{
   if (val > 1023) val = 1023;
   CCPR1L = (val >> 2) & 0xFF;
   CCP1CONbits.DC1B = val & 0x03;
```

```
    CCPR2L = (val >> 2) & 0xFF;
    CCP2CONbits.DC2B = val & 0x03;
}

/* System initialization */
void init_system(void)
{
    TRISA = 0xFF;   // PORTA all inputs (HC-SR04)
    PORTA = 0x00;
    ANSELA = 0x00;  // Digital I/O
    init_I2C();
    init_PWM();
    /* Additional init: interrupts, AI module, etc. */
}
```