German University in Cairo

Mechatronics Engineering (MCTR601)

Object Classifying Robotic Arm

Name	ID	Lab Number
Ahmed Husseini	58-21270	T38
Mohamed Yaser	58-21125	T35
Mohamed Mohsen	58-1653	T36

Table of Contents

1. Project Description

- Generally explain your project idea and objectives and applications.
- Include a concise overview of what the device does and how it works.
- Mention the main actuators and sensors of the system
- A functional diagram showing all major components and their connections.
- Be sure to label key components in your figures (with concise text and arrows).

Part number	Name
1	Base
2	Arm
3	Gripplers
4	Servo Motors

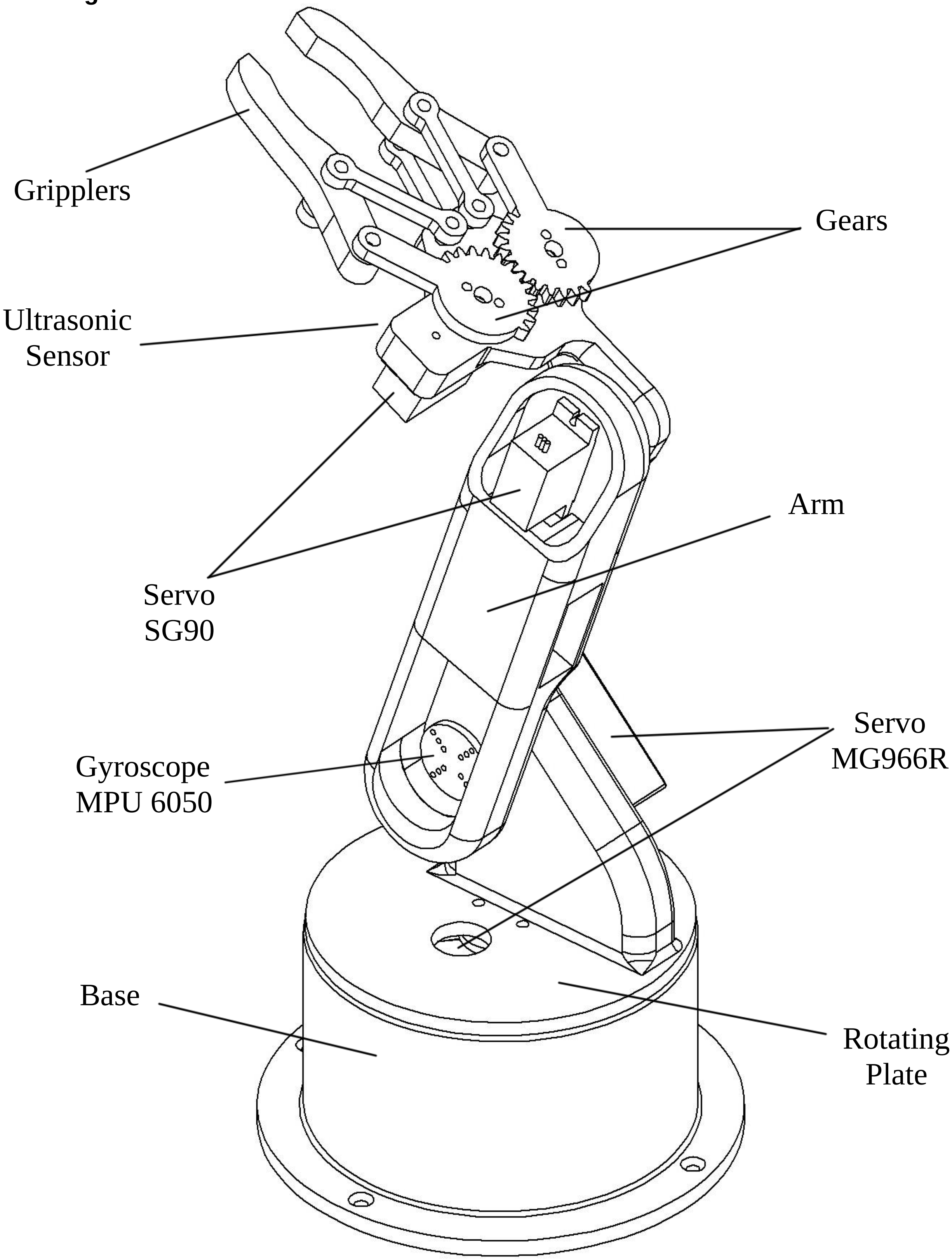
Overview:
This project involves designing and implementing a simple robotic arm capable of classifying and sorting objects based on their color. The robot picks up randomly placed balls from a central basket and places them into designated places based on their color.

Working Principle:
1. The robotic arm picks up a ball from the central basket.
2. A color detection sensor determines whether the ball is blue or red.
3. Based on the detected color, the arm moves and places the ball into the corresponding basket: Blue ball → Right basket
Red ball → Left basket
4. The process repeats for each ball until the central basket is empty.

Applications:
- Industry and Automation: This project simulates basic industrial product sorting operations in factories.
- Medical Field: Can be used to classify medicines or blood samples in hospitals or pharmacies.
- AI & Machine Learning Integration: Can be expanded by integrating machine learning for improved object recognition.

Actuators: 2x Servo Motors + 2x Micro Servo Motors
Sensors: Ultrasonic (HC-SR04) + Color Sensor (TCS3200) + Gyro (MPU6050)
Modules: Servo I2C Driver (PCA9685) + I2C LCD Screen

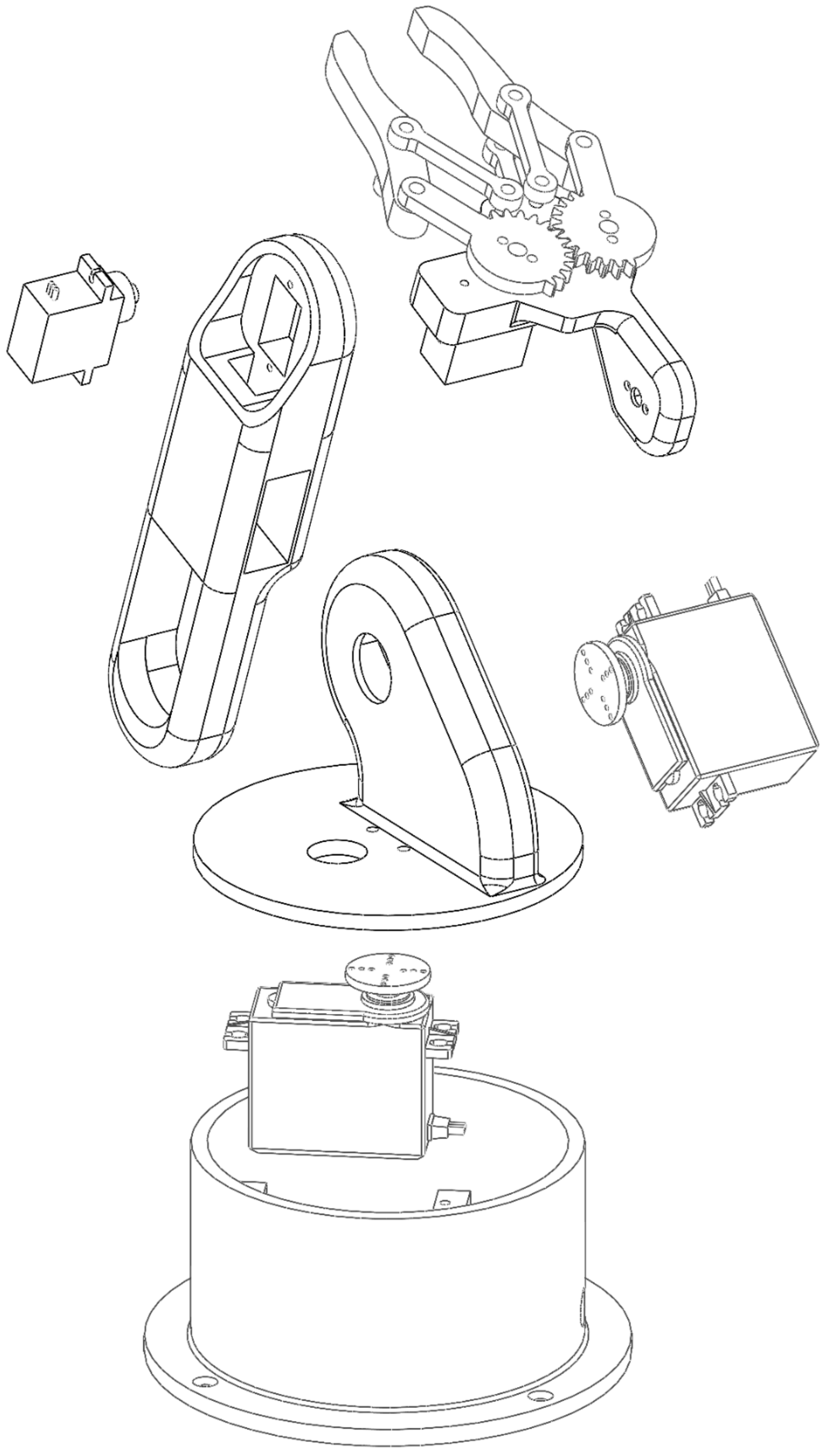
Diagram:

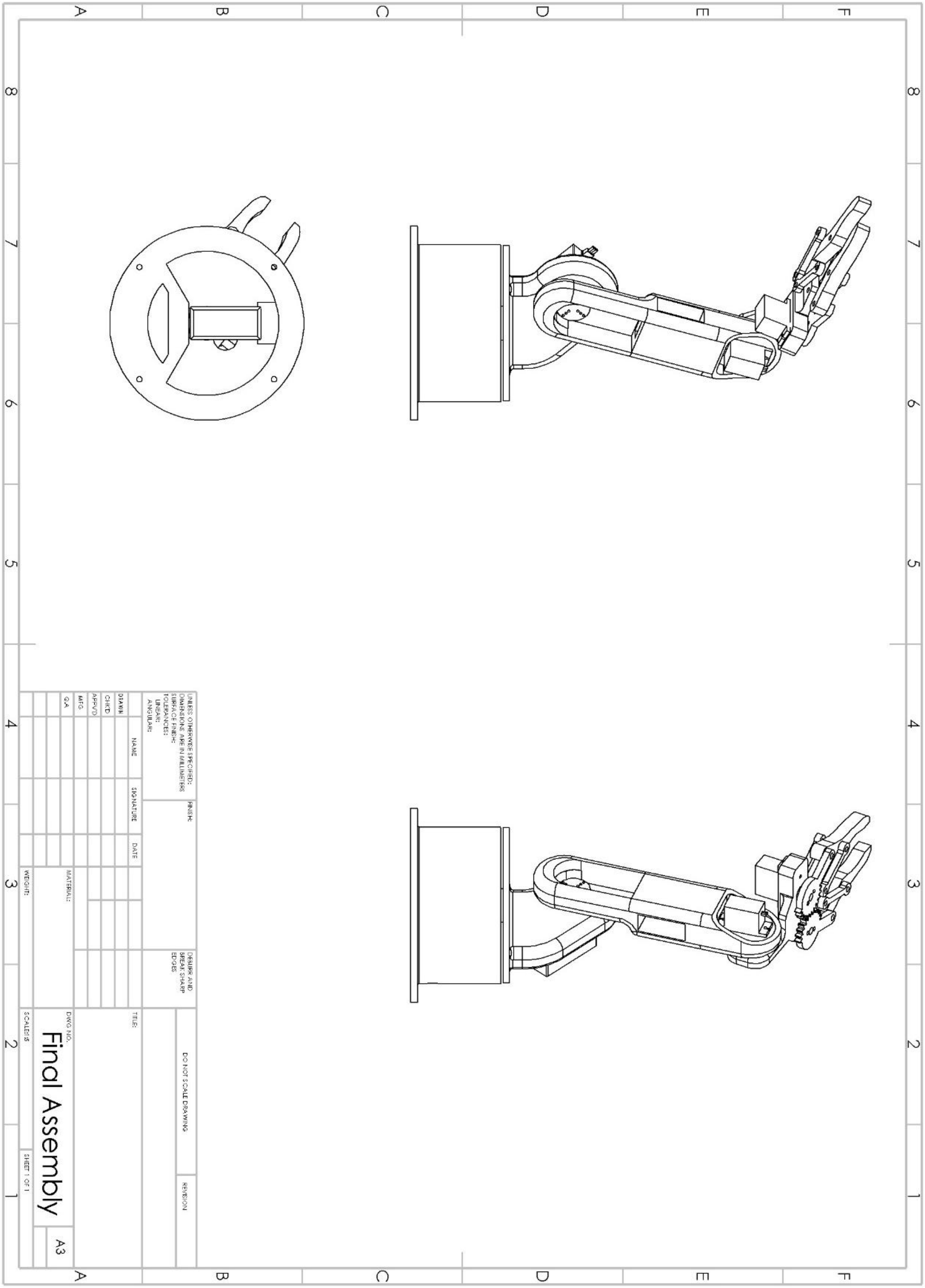

Gripplers
Gears
Ultrasonic
Sensor
Arm
Servo
SG90
Servo
MG966R
Gyroscope
MPU 6050
Base
Rotating
Plate

2. Methodology

2.1 Mechanical design

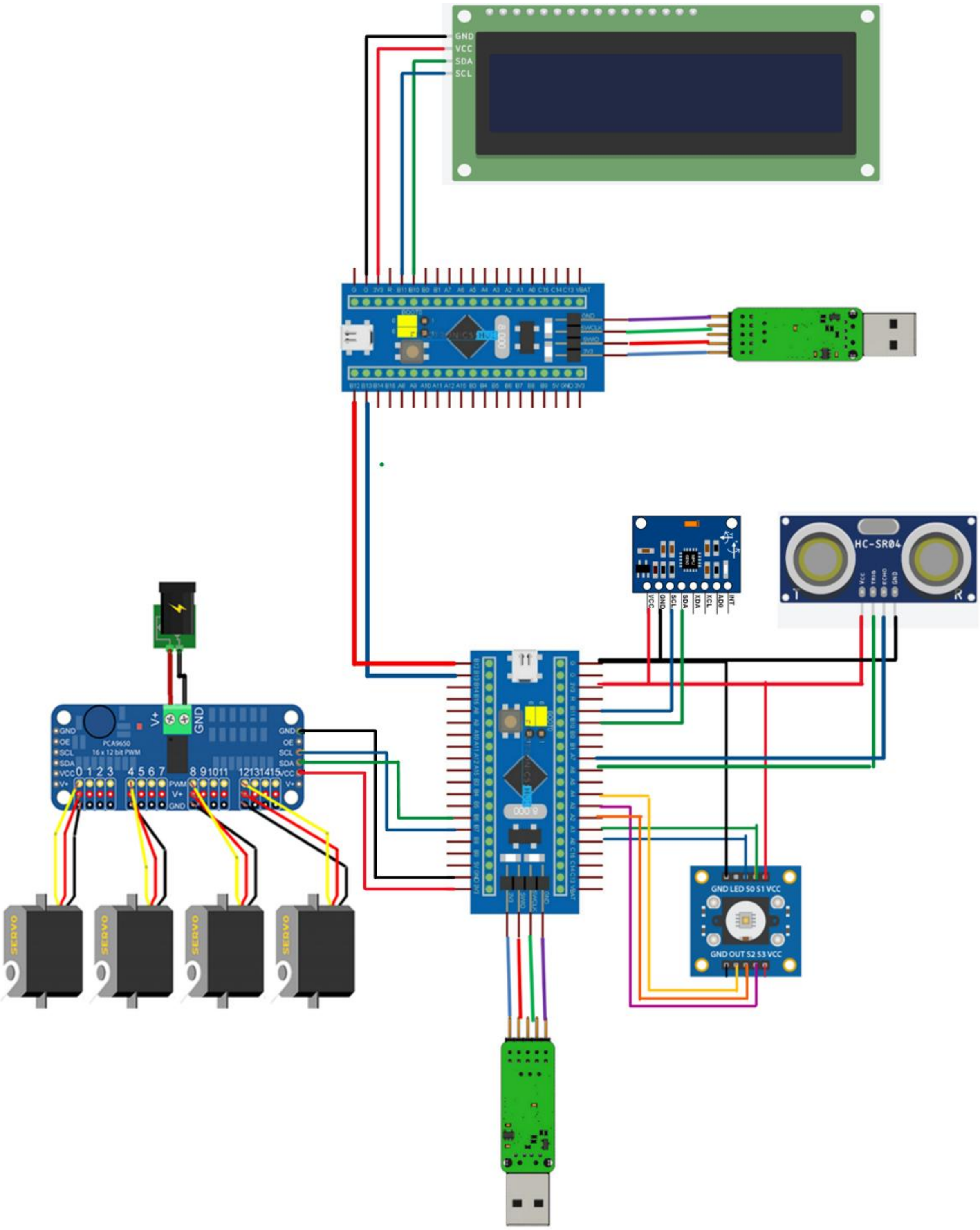
Include a detailed description of the system's mechanical and electromechanical components




UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN MILLIMETERS
SURFACE FINISH:
TOLERANCES:
LINEAR:
ANGULAR:
FINISH:
DEBURR AND BREAK SHARP EDGES
DO NOT SCALE DRAWING
REVISION
NAME
SIGNATURE
DATE
DRAWN
CHK'D
APPV'D
MFG
Q.A
MATERIAL:
WEIGHT:
TITLE:
DWG NO.
Final Assembly
A3
SCALE:1:5
SHEET 1 OF 1

2.2 Electrical design

Include schematics for the electrical systems showing all connections



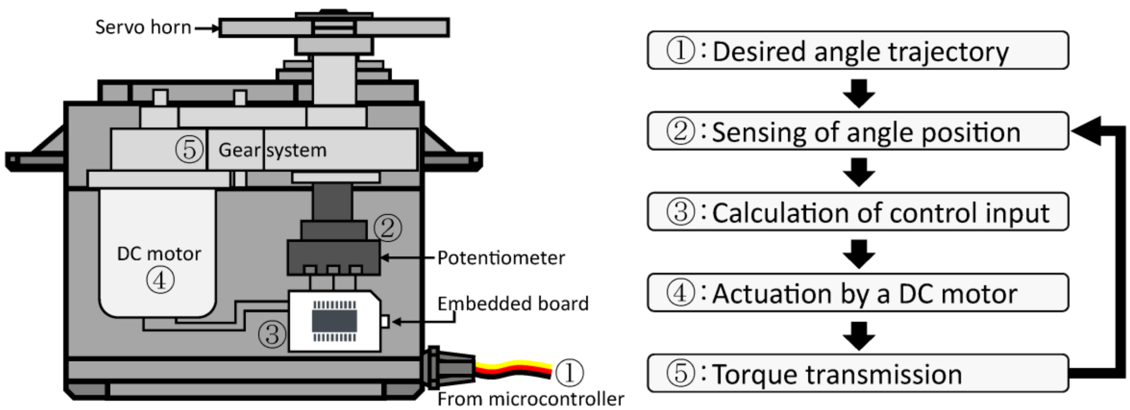
Control

2.3.1. Modeling

This section should show the

- **Derivation and linearization of the governing equations**
- **Derivation of the Transfer function model of the system**
- **Block Diagram representation**

The controlled variable in this project is the motor’s angular position. And since Servo motors already come with a closed-loop control, we hacked it and used the DC motor and the potentiometer individually.



After simulating the mechanical model of the arm Solidworks, we found that:

$Moment\ of\ Inertia\ (J) =\ 0.01\ kg.m^2$

$Damping\ Coefficient\ (b) =\ 0.001\ N.m.s$

And, from the motor’s datasheet:

$Motor\ Constant\ (K) =\ 0.05\ N.m/A$

$Armature\ Resistance\ (R)\ =\ 2\ \Omega$

$Armature\ Inductance\ (L)\ =\ 0.005\ H$

a) The mathematical model of the DC motor system:

$$v_i - iR - L\frac{di}{dt} - K_e\dot{\theta}_m = 0 \tag{1}$$

$$J_m\ddot{\theta}_m = T - B\dot{\theta}_m \qquad \Longrightarrow \qquad J_m\ddot{\theta}_m + B\dot{\theta}_m = K_T i \tag{2}$$

$$J_L\ddot{\theta} = \frac{1}{n}T \qquad \Longrightarrow \qquad J_L\ddot{\theta} = \frac{1}{n}K_T i \tag{3}$$

b) Obtaining transfer function:

Using Laplace Transform:

$$Eq1: \quad V_i(s) - RI(s) - LsI(s) - K_e s\Theta_m(s) = 0$$

$$V_i(s) = (R + Ls)\, I(s) + K_e s\Theta_m(s) \tag{1}$$

$$Eq2: \quad J_m s^{2}\Theta_m(s) + Bs\Theta_m(s) = K_T I(s)$$

$$\Theta_m(s) = \frac{K_T}{J_m s^{2} + Bs}\, I(s) \tag{2}$$

$$Eq3: \quad J_L s^{2}\Theta(s) - \frac{1}{n} K_T I(s) = 0$$

$$I(s) = \frac{nJ_L s^{2}}{K_T}\, \Theta(s) \tag{3}$$

from (2) and (3) we get : $\Theta_m(s) = \frac{K_T}{J_m s^{2}+Bs} \times \frac{nJ_L s^{2}}{K_T}\, \Theta(s) = \frac{nJ_L s^{2}}{J_m s^{2}+Bs}\, \Theta(s)$ (4)

substituting with (3) and (4) in (1) :

$$V_i(s) = (R + Ls)\frac{nJ_L s^{2}}{K_T}\, \Theta(s) + K_e s \frac{nJ_L s^{2}}{J_m s^{2} + Bs}\, \Theta(s)$$

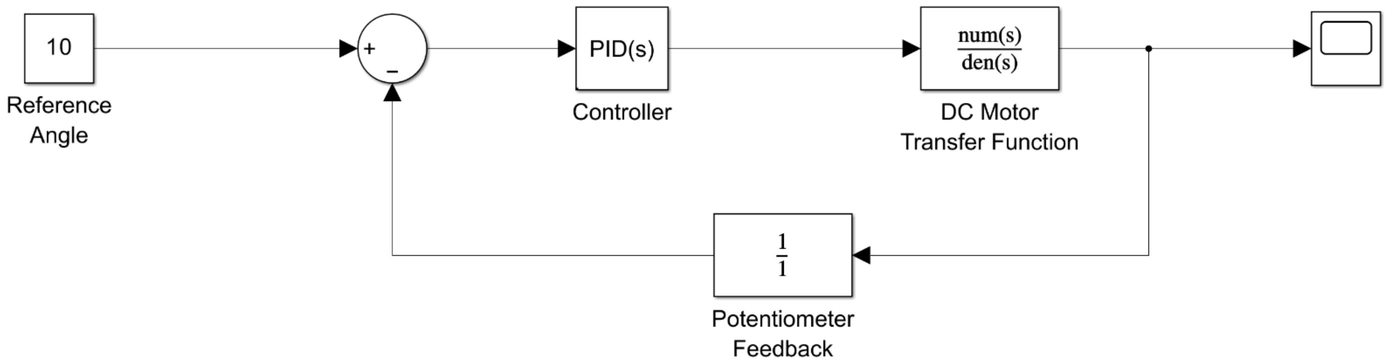
$$V_i(s) = \left(\frac{RnJ_L s^{2} + LnJ_L s^{3}}{K_T} + \frac{K_e n J_L s^{3}}{J_m s^{2} + Bs}\right)\Theta(s)$$

$$V_i(s) = \frac{nJ_L \mathrm{s}((J_m \mathrm{s} + \mathrm{B})(\mathrm{Ls} + \mathrm{R}) + K_T K_e)}{J_m K_T}\, \Theta(s)$$

$$\frac{\Theta(s)}{V_i(s)} = \frac{J_m}{\mathrm{n}J_L}\frac{K_T}{\mathrm{s}((J_m \mathrm{s} + \mathrm{B})(\mathrm{Ls} + \mathrm{R}) + K_T K_e)}$$

$$\boxed{\boldsymbol{Transfer\ Function} = \frac{\mathbf{0.01}}{\mathbf{10}}\frac{\mathbf{0.05}}{\mathbf{s((0.01\,s + 0.001)(0.005\,s + 2) + 0.05)}}}$$

c) Simulink Block diagram



2.3.2. Analysis

- **Solve equations and get the open loop response**

Step response $\boldsymbol{h(t) = L^{-1}\left\{H(s) * \frac{1}{s}\right\}}$

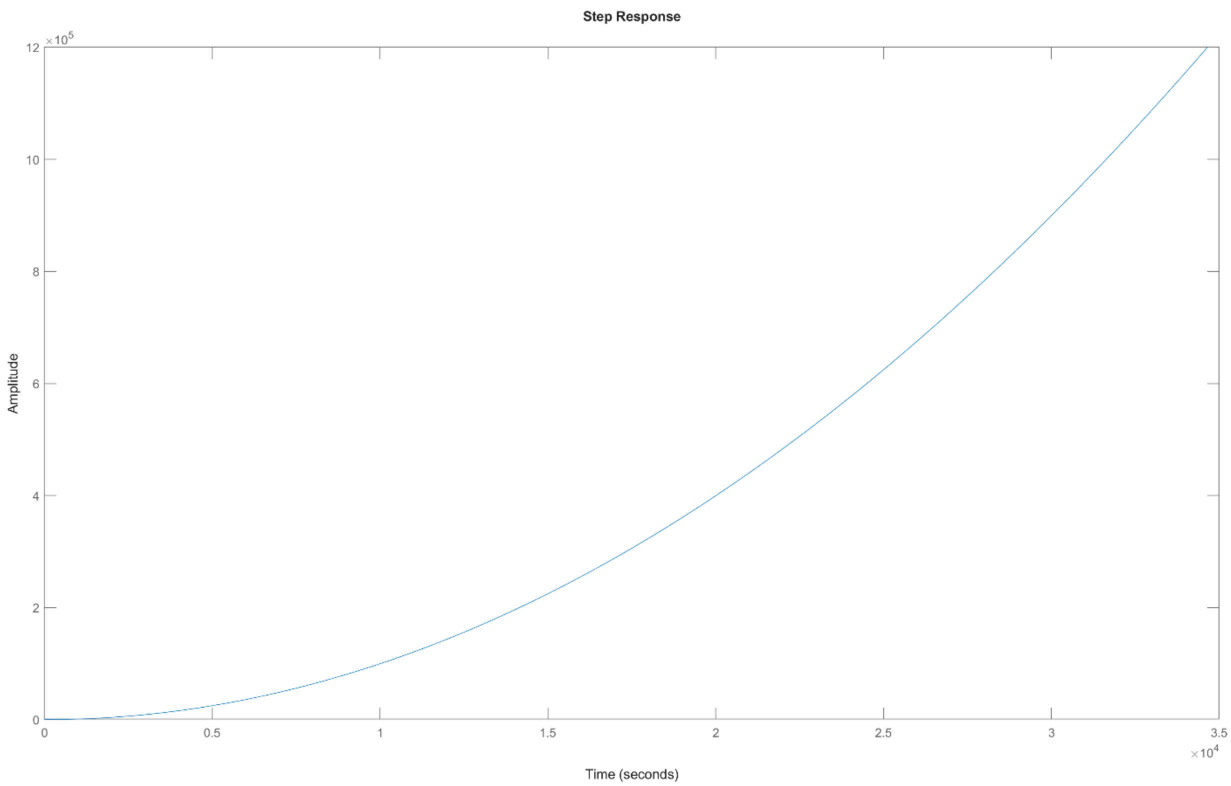
$$H(s) = \frac{J_m K_T}{\mathrm{n} J_L} \frac{1}{\mathrm{s}\big((J_m \mathrm{s} \ + \ \mathrm{B})(\mathrm{Ls} \ + \ \mathrm{R}) + \ K_T K_e\big)} * \frac{1}{s}$$

$$H(s) = \frac{0.01}{10} \frac{0.05}{\mathrm{s}^2((0.01 \ \mathrm{s} \ + \ 0.001)(0.005 \ \mathrm{s} \ + \ 2) \ + \ 0.05)}$$

$$H(s) = \frac{0.0005}{s^2(0.005 \ s^2 + 2.0005 \ s + 0.25)}$$

$$H(s) = \frac{6.188 \times 10^{-5}}{\mathrm{s}} + \frac{6.188 \times 10^{-5}}{\mathrm{s} + 3.125} + \frac{6.188 \times 10^{-5}}{\mathrm{s} + 1374.38}$$

$$h(t) = 6.188 \times 10^{-5} - 6.188 \times 10^{-5} e^{-3.125\mathrm{t}} - 6.188 \times 10^{-5} e^{-1374.38\mathrm{t}}$$



2.3.3. Controller Design

- Show the controller design steps
- Clearly show the control law
- Show the closed loop simulations

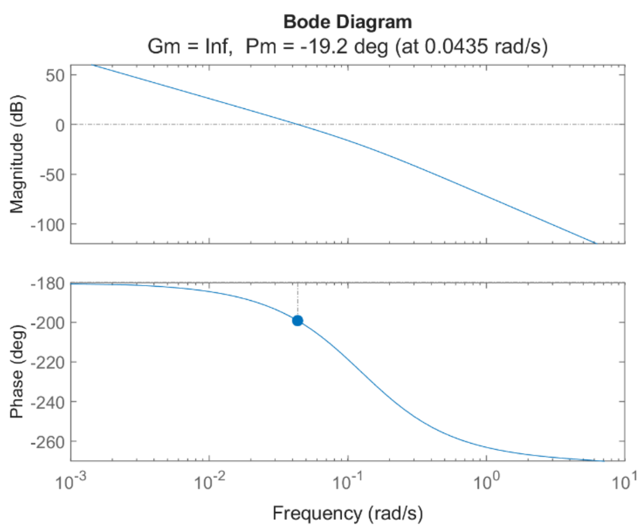
Hint: You may use MATLAB's SISO tool for the simulations in the analysis and design stages

Step 1: Define the Plant in MATLAB

```
num = 0.0005;
den = conv([1 0 0], [0.005 2.0005 0.25]);  % Multiply s^2 with the quadratic
G = tf(num, den);
```

Step 2: Analyze Open-Loop System

```
bode(G)
margin(G)
step(G)
```



Step 3: Design PID Controller Using `pidtune` Function

```
C = pidtune(G, 'PID');  % Automatically tunes a PID controller
C = pid(Kp, Ki, Kd);
```

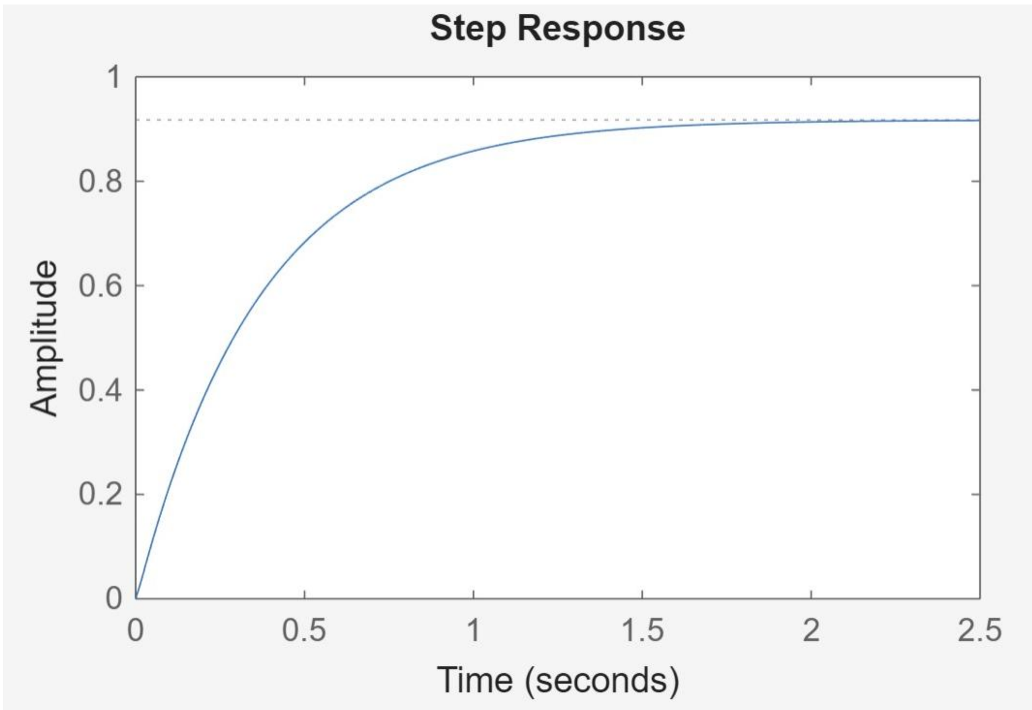
$$C = K_p + K_d s + \frac{K_i}{s} = 50 + 2\,s + \frac{100}{s} = \frac{50\,s + 2\,s^2 + 100}{s}$$

Step 4: Form the Closed-Loop System

```
T = feedback(C*G, 1);  % Unity feedback
```

Step 5: Analyze Closed-Loop Response

```
step(T)
```



2.3 Programming

Include the used codes and a concise software flowchart

```
#include "stm32f103xb.h"
#include "FreeRTOS.h"
#include "task.h"
#include <stdio.h>

// ==== PIN DEFINITIONS ====

// Ultrasonic

#define TRIG_PIN     6   // PA6
#define ECHO_PIN     7   // PA7
#define DEBUG_PIN    14  // PB14

// Color sensor

#define LED_RED_PIN  12  // PB12
#define LED_BLUE_PIN 13  // PB13

// PCA9685

#define PCA9685_ADDRESS         0x80
#define PCA9685_MODE1           0x00
#define PCA9685_PRE_SCALE       0xFE
#define PCA9685_LED0_ON_L       0x06
#define PCA9685_MODE1_SLEEP_BIT 4
#define PCA9685_MODE1_AI_BIT    5
#define PCA9685_MODE1_RESTART_BIT 7

// ==== GLOBAL VARIABLES ====

volatile float distance_cm = 0;
volatile uint16_t capture1 = 0, capture2 = 0;
volatile uint8_t captured = 0;

// ==== UTILS ====

void delay_us(uint16_t us) {
    TIM1->CNT = 0;
    while (TIM1->CNT < us);
}

void delayMs(uint32_t ms) {
    vTaskDelay(pdMS_TO_TICKS(ms));
}



// ==== CLOCK CONFIG ====
```

```c
void sys_clock_config(void) {
    RCC->CR |= RCC_CR_HSEON;
    while (!(RCC->CR & RCC_CR_HSERDY));
    RCC->CFGR |= RCC_CFGR_PLLSRC | RCC_CFGR_PLLMULL9;
    RCC->CR |= RCC_CR_PLLON;
    while (!(RCC->CR & RCC_CR_PLLRDY));
    FLASH->ACR |= FLASH_ACR_LATENCY_2;
    RCC->CFGR |= RCC_CFGR_SW_PLL;
    while (!(RCC->CFGR & RCC_CFGR_SWS_PLL));
    SystemCoreClockUpdate();
}

// ==== GPIO INIT ====

void gpio_init(void) {
    RCC->APB2ENR |= RCC_APB2ENR_IOPAEN | RCC_APB2ENR_IOPBEN |
    RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;

    // PA6 – TRIG
    GPIOA->CRL &= ~(0xF << (4 * TRIG_PIN));
    GPIOA->CRL |=  (0x1 << (4 * TRIG_PIN));

    // PA7 – ECHO
    GPIOA->CRL &= ~(0xF << (4 * ECHO_PIN));
    GPIOA->CRL |=  (0x4 << (4 * ECHO_PIN));

    // PB14 - Debug LED
    GPIOB->CRH &= ~(0xF << (4 * (DEBUG_PIN - 8)));
    GPIOB->CRH |=  (0x1 << (4 * (DEBUG_PIN - 8)));

    // PA0 - TCS3200 OUT (input floating)
    GPIOA->CRL &= ~(GPIO_CRL_MODE0 | GPIO_CRL_CNF0);
    GPIOA->CRL |= GPIO_CRL_CNF0_0;

    // PA1–PA4 - S0–S3 (outputs)
    GPIOA->CRL |= (GPIO_CRL_MODE1_1 | GPIO_CRL_MODE2_1 |
    GPIO_CRL_MODE3_1 | GPIO_CRL_MODE4_1);



    // PB12, PB13 – LEDs
    GPIOB->CRH &= ~(GPIO_CRH_MODE12 | GPIO_CRH_CNF12 |
    GPIO_CRH_MODE13 | GPIO_CRH_CNF13);
    GPIOB->CRH |= (GPIO_CRH_MODE12_1 | GPIO_CRH_MODE13_1);

    // PB6, PB7 - I2C1 (alt function open-drain)
    GPIOB->CRL &= ~(GPIO_CRL_MODE6 | GPIO_CRL_CNF6 |
    GPIO_CRL_MODE7 | GPIO_CRL_CNF7);
    GPIOB->CRL |= (GPIO_CRL_MODE6_1 | GPIO_CRL_CNF6_1 | GPIO_CRL_CNF6_0);
    GPIOB->CRL |= (GPIO_CRL_MODE7_1 | GPIO_CRL_CNF7_1 | GPIO_CRL_CNF7_0);
}
```

```c
// ==== TIM CONFIG ====

void TIM1_us_init(void) {
    RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;
    TIM1->PSC = (SystemCoreClock / 1000000) - 1;
    TIM1->ARR = 0xFFFF;
    TIM1->CR1 |= TIM_CR1_CEN;
}

void TIM2_InputCapture_Config(void) {
    RCC->APB1ENR |= RCC_APB1ENR_TIM2EN;
    TIM2->PSC = 71;
    TIM2->ARR = 0xFFFF;
    TIM2->CCMR1 = 0x01;
    TIM2->CCER = TIM_CCER_CC1E;
    TIM2->DIER |= TIM_DIER_CC1IE;
    TIM2->CR1 |= TIM_CR1_CEN;
    NVIC_EnableIRQ(TIM2_IRQn);
}

// ==== INTERRUPTS ====

void TIM2_IRQHandler(void) {
    if (TIM2->SR & TIM_SR_CC1IF) {
        if (!captured) capture1 = TIM2->CCR1, captured = 1;
    else capture2 = TIM2->CCR1, captured = 2;
    TIM2->SR &= ~TIM_SR_CC1IF;
    }
}

// ==== ULTRASONIC ====

float measure_distance(void) {
    GPIOA->BSRR = (1 << TRIG_PIN);
    delay_us(10);
    GPIOA->BRR = (1 << TRIG_PIN);

    while (!(GPIOA->IDR & (1 << ECHO_PIN)));
    uint32_t start = TIM1->CNT;
    while (GPIOA->IDR & (1 << ECHO_PIN));
    uint32_t end = TIM1->CNT;

    uint32_t diff = (end >= start) ? (end - start) : (0xFFFF - start + end);
    return (diff * 0.0343f) / 2.0f;
}
```

```c
// ==== TCS3200 COLOR SENSOR ====

void set_color_filter(uint8_t s2, uint8_t s3) {
    if (s2) GPIOA->BSRR = GPIO_BSRR_BS3; else GPIOA->BSRR = GPIO_BSRR_BR3;
    if (s3) GPIOA->BSRR = GPIO_BSRR_BS4; else GPIOA->BSRR = GPIO_BSRR_BR4;
}

uint32_t measure_frequency(void) {
    captured = 0;
    while (captured < 2);
    uint16_t period_ticks = (capture2 > capture1) ? (capture2 - capture1) :
    (0xFFFF - capture1 + capture2);
    return (1000000UL / period_ticks);
}

// ==== I2C & PCA9685 ====

void I2C1_Init(void) {
    RCC->APB1ENR |= RCC_APB1ENR_I2C1EN;
    I2C1->CR1 &= ~I2C_CR1_PE;
    I2C1->CR2 = 36;
    I2C1->CCR = 180;
    I2C1->TRISE = 37;
    I2C1->CR1 |= I2C_CR1_PE;
}

void I2C1_Write(uint8_t devAddr, uint8_t regAddr, uint8_t *data, uint8_t len){
    while (I2C1->SR2 & I2C_SR2_BUSY);
    I2C1->CR1 |= I2C_CR1_START;
    while (!(I2C1->SR1 & I2C_SR1_SB)); (void)I2C1->SR1;
    I2C1->DR = devAddr;
    while (!(I2C1->SR1 & I2C_SR1_ADDR)); (void)I2C1->SR2;
    I2C1->DR = regAddr;
    while (!(I2C1->SR1 & I2C_SR1_TXE));

    for (uint8_t i = 0; i < len; i++) {
        I2C1->DR = data[i];
        while (!(I2C1->SR1 & I2C_SR1_TXE));
    }
    I2C1->CR1 |= I2C_CR1_STOP;
}
```

```
void I2C1_Read(uint8_t devAddr, uint8_t regAddr, uint8_t *data, uint8_t len) {
    while (I2C1->SR2 & I2C_SR2_BUSY);
    I2C1->CR1 |= I2C_CR1_START;
    while (!(I2C1->SR1 & I2C_SR1_SB)); (void)I2C1->SR1;
    I2C1->DR = devAddr;
    while (!(I2C1->SR1 & I2C_SR1_ADDR)); (void)I2C1->SR2;
    I2C1->DR = regAddr;
    while (!(I2C1->SR1 & I2C_SR1_TXE));
    I2C1->CR1 |= I2C_CR1_START;
    while (!(I2C1->SR1 & I2C_SR1_SB)); (void)I2C1->SR1;
    I2C1->DR = devAddr | 0x01;
    while (!(I2C1->SR1 & I2C_SR1_ADDR)); (void)I2C1->SR2;
    for (uint8_t i = 0; i < len; i++) {
        if (i == len - 1) I2C1->CR1 &= ~I2C_CR1_ACK;
        while (!(I2C1->SR1 & I2C_SR1_RXNE));
        data[i] = I2C1->DR;
    }
    I2C1->CR1 |= I2C_CR1_STOP;
    I2C1->CR1 |= I2C_CR1_ACK;
}

void PCA9685_SetBit(uint8_t reg, uint8_t bit, uint8_t value) {
    uint8_t data;
    I2C1_Read(PCA9685_ADDRESS, reg, &data, 1);
    data = (value == 0) ? (data & ~(1 << bit)) : (data | (1 << bit));
    I2C1_Write(PCA9685_ADDRESS, reg, &data, 1);
    delayMs(1);
}

void PCA9685_SetPWMFrequency(uint16_t freq) {
    uint8_t prescale = (freq >= 1526) ? 3 : (freq <= 24) ? 255 :
    (uint8_t)(25000000 / (4096 * freq));
    PCA9685_SetBit(PCA9685_MODE1, PCA9685_MODE1_SLEEP_BIT, 1);
    I2C1_Write(PCA9685_ADDRESS, PCA9685_PRE_SCALE, &prescale, 1);
    PCA9685_SetBit(PCA9685_MODE1, PCA9685_MODE1_SLEEP_BIT, 0);
    PCA9685_SetBit(PCA9685_MODE1, PCA9685_MODE1_RESTART_BIT, 1);
}

void PCA9685_SetPWM(uint8_t ch, uint16_t on, uint16_t off) {
    uint8_t data[4] = { on & 0xFF, on >> 8, off & 0xFF, off >> 8 };
    I2C1_Write(PCA9685_ADDRESS, PCA9685_LED0_ON_L + 4 * ch, data, 4);
}

void PCA9685_SetServoAngle(uint8_t ch, float angle) {
    float val = (angle * (511.9 - 102.4) / 180.0f) + 102.4f;
    PCA9685_SetPWM(ch, 0, (uint16_t)val);
}
```

```c
// ==== TASKS ====

void SensorTask(void* arg) {
    while (1) {
        distance_cm = measure_distance();
        if (distance_cm < 10.0f)
        GPIOB->BSRR = (1 << DEBUG_PIN);
    else
        GPIOB->BRR = (1 << DEBUG_PIN);
        vTaskDelay(pdMS_TO_TICKS(200));
    }
}

void vColorTask(void *pv) {
    uint32_t freq_r, freq_b;

    while (1) {

        // Red

        set_color_filter(0, 0); // S2=0, S3=0 => Red
        vTaskDelay(pdMS_TO_TICKS(100));
        freq_r = measure_frequency();

        // Blue

        set_color_filter(0, 1); // S2=0, S3=1 => Blue
        vTaskDelay(pdMS_TO_TICKS(100));
        freq_b = measure_frequency();

        // Determine dominant color and light LEDs

        if (freq_r > freq_b) {
            GPIOB->BSRR = GPIO_BSRR_BS12;                  // Red LED ON
            GPIOB->BSRR = GPIO_BSRR_BR13;                  // Blue LED OFF
        } else if (freq_b > freq_r) {
            GPIOB->BSRR = GPIO_BSRR_BS13;                  // Blue LED ON
            GPIOB->BSRR = GPIO_BSRR_BR12;                  // Red LED OFF
        } else {
            GPIOB->BSRR = GPIO_BSRR_BR12 | GPIO_BSRR_BR13; // Both OFF
        }

        vTaskDelay(pdMS_TO_TICKS(300));
    }
}
```

```c
void vServoTask(void *pv) {
    I2C1_Init(); PCA9685_SetPWMFrequency(50);
    PCA9685_SetBit(PCA9685_MODE1, PCA9685_MODE1_AI_BIT, 1);
    while (1) {
        PCA9685_SetServoAngle(0, 90);
        PCA9685_SetServoAngle(1, 30);
        PCA9685_SetServoAngle(2, 120);
        delayMs(1000);
        PCA9685_SetServoAngle(3, 180);
        delayMs(1000);

        PCA9685_SetServoAngle(1, 130);
        PCA9685_SetServoAngle(2, 90);
        delayMs(1000);
        while ((GPIOB->IDR & GPIO_IDR_IDR14)==0);
        PCA9685_SetServoAngle(3, 30);
        delayMs(1000);

        PCA9685_SetServoAngle(1, 30); // Color Measuring Position
        delayMs(3000);

        while(((GPIOB->IDR & GPIO_IDR_IDR12)==0)
        & ((GPIOB->IDR & GPIO_IDR_IDR13)==0));

        if (GPIOB->IDR & GPIO_IDR_IDR12) {
            PCA9685_SetServoAngle(0, 180);
        } else {
        PCA9685_SetServoAngle(0, 0);
        }

        PCA9685_SetServoAngle(1, 100);
        PCA9685_SetServoAngle(2, 90);
        delayMs(1000);
        PCA9685_SetServoAngle(3, 180);
        delayMs(1000);
    }
}

// ==== MAIN ====

int main(void) {
    sys_clock_config();
    gpio_init();
    TIM1_us_init();
    TIM2_InputCapture_Config();
    GPIOA->BSRR = GPIO_BSRR_BS1 | GPIO_BSRR_BS2;  // S0, S1 = 1

    xTaskCreate(SensorTask, "Sensor", 128, NULL, 2, NULL);
    xTaskCreate(vColorTask, "Color", 256, NULL, 1, NULL);
    xTaskCreate(vServoTask, "Servo", 256, NULL, 1, NULL);
    vTaskStartScheduler();

    while (1);
}
```

3. Design Evaluation

Briefly describe the success of the hardware by comparing the experimental results to the model simulations.
Comment on the obtained results of the system performance and the graphs.

Mechanical

In the Mechanical part, everything worked perfectly as designed. The robot **members** were able to bare the stress as calculated in the stress analysis part. All the **motors** generated enough torque to move and rotate their respective member.

Electrical

In the Electrical part, the performance was satisfying.

4. Appendix

Append detailed wiring diagrams (if details are not included in earlier figures), anything else supporting the system details section.