Design of a holonomic five legged robot

Final Report

L. Steyn
04496486

Submitted as partial fulfilment of the requirements of Project EPR400
in the Department of Electrical, Electronic and Computer Engineering
University of Pretoria

November 2017

Study leader: Dr. J.D. Le Roux

Description of product

This report contains the technical documentation required to design a holonomic five legged robot. The robot is built using 15 servo motors, 3D printed parts, a 32 bit microcontroller and a smartphone application that functions as a user interface and remote control.

TABLE OF CONTENTS

1. System block

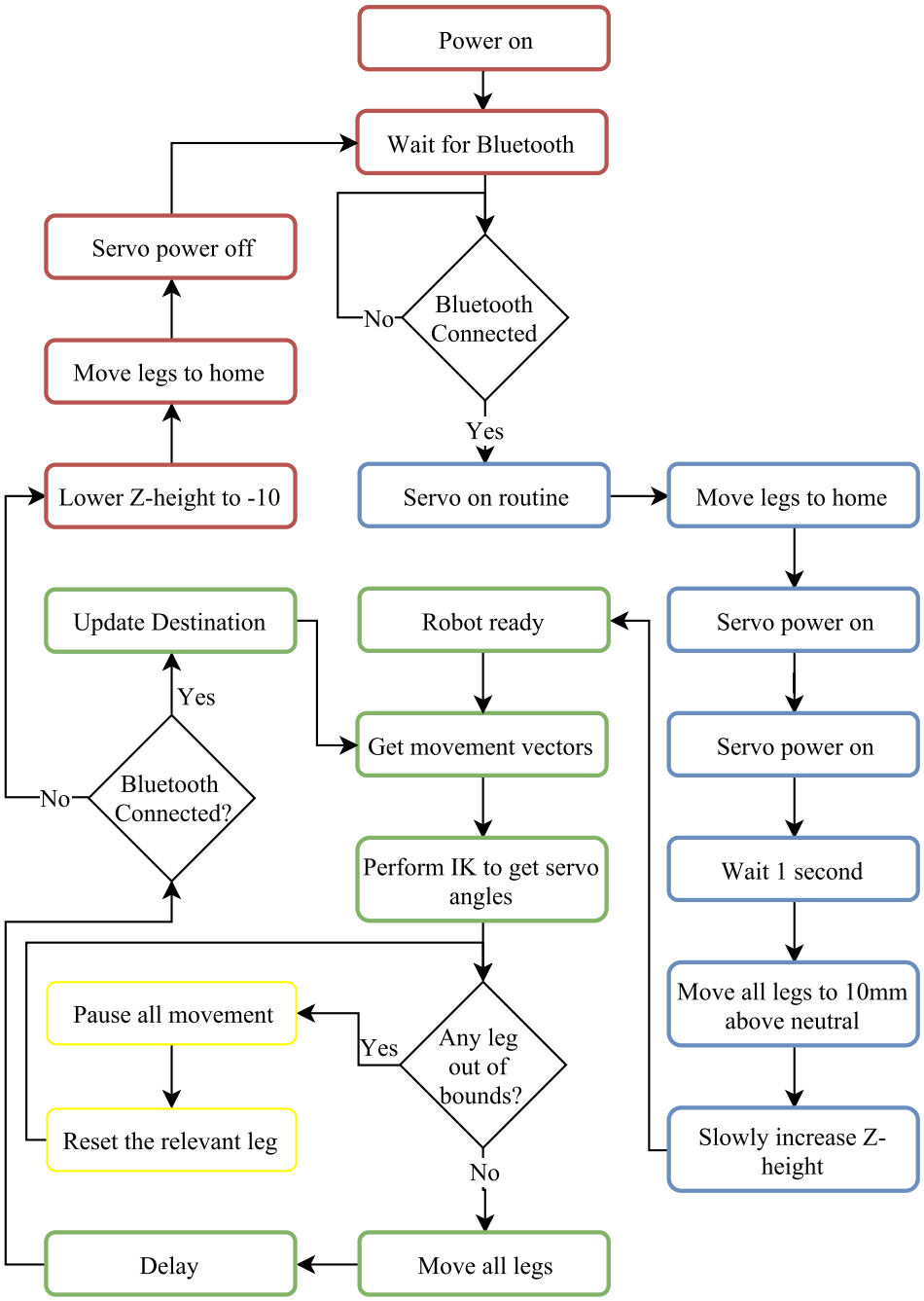

Figure 1. System block diagram.

2. Systems level description of design

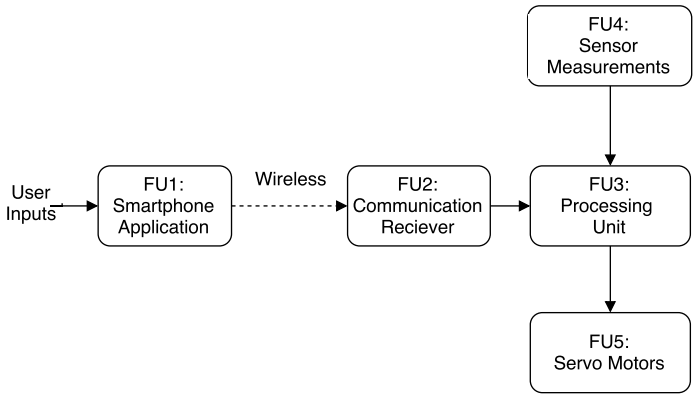

Figure 2. System level design.

3. Block diagrams of modules

The design does not consist of modules. The system block diagram can be seen above

4. Description of modules

The design does not consist of modules. The complete description of the design can be found in part 4 of this report.

5. Description of interfacing with other modules

The only interfacing in this project is between the smartphone and the robot. More details on this can be found in part 4 of this report

6. Complete circuit diagram

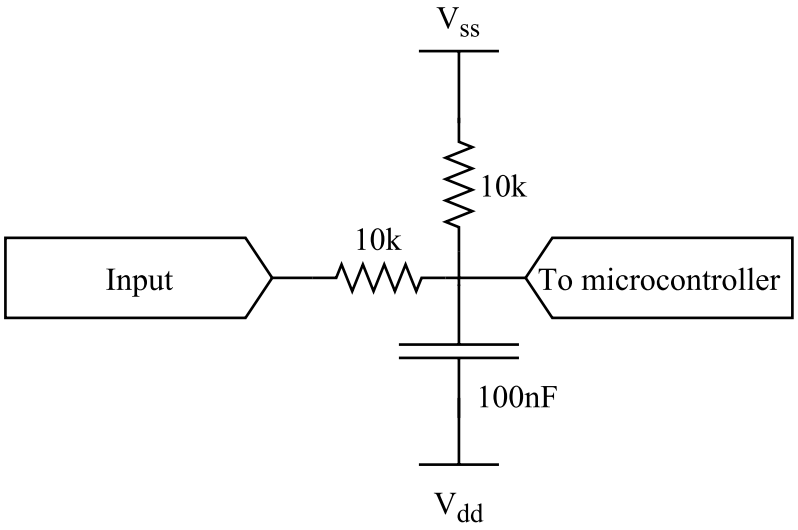

Figure 3. Input protection design.

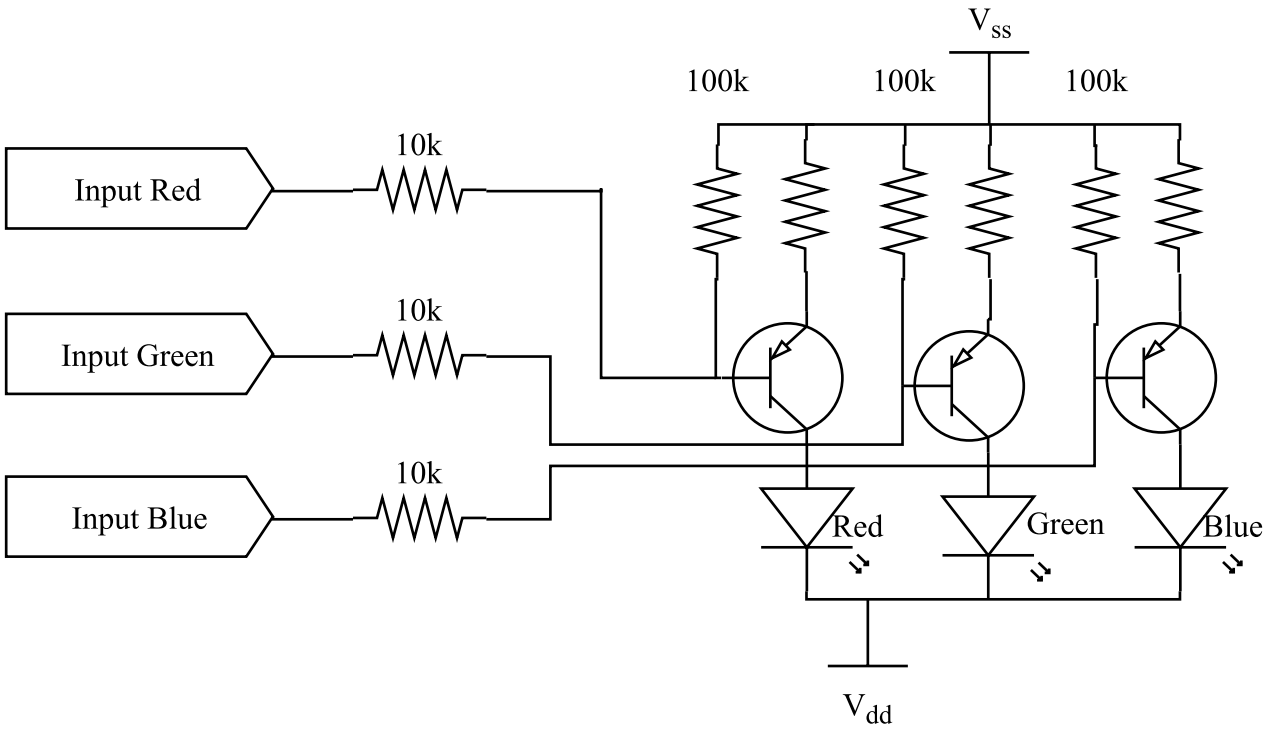

Figure 4. LED driver design.

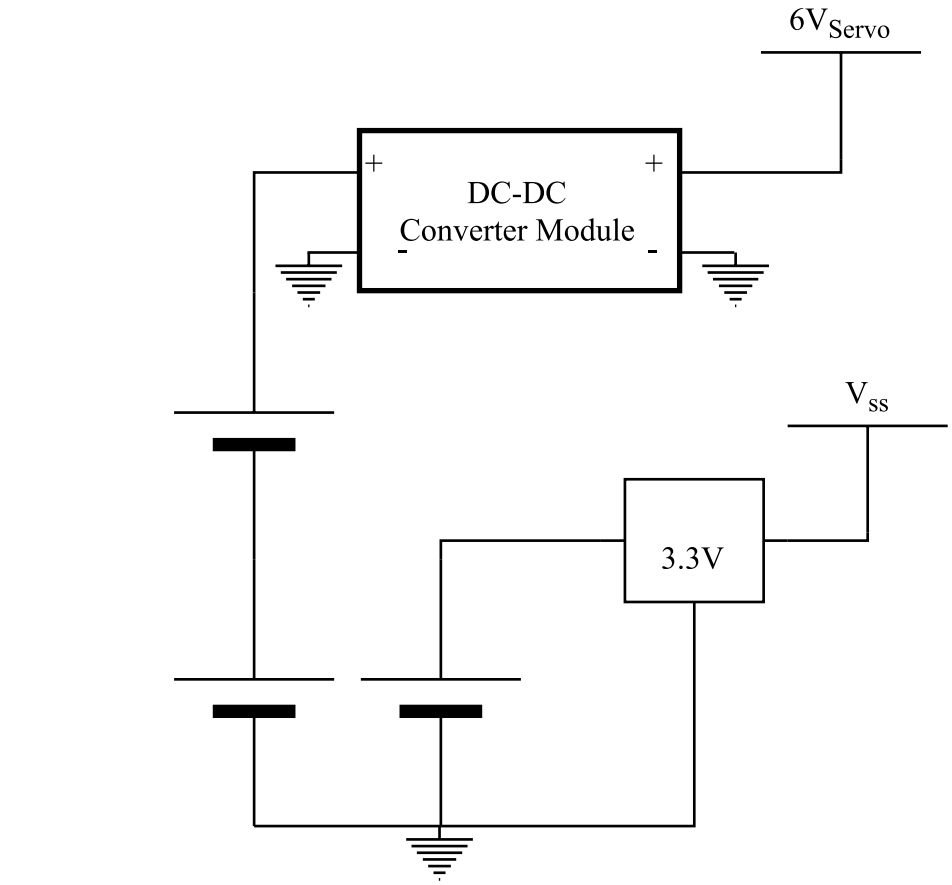

Figure 5. Power supply design.

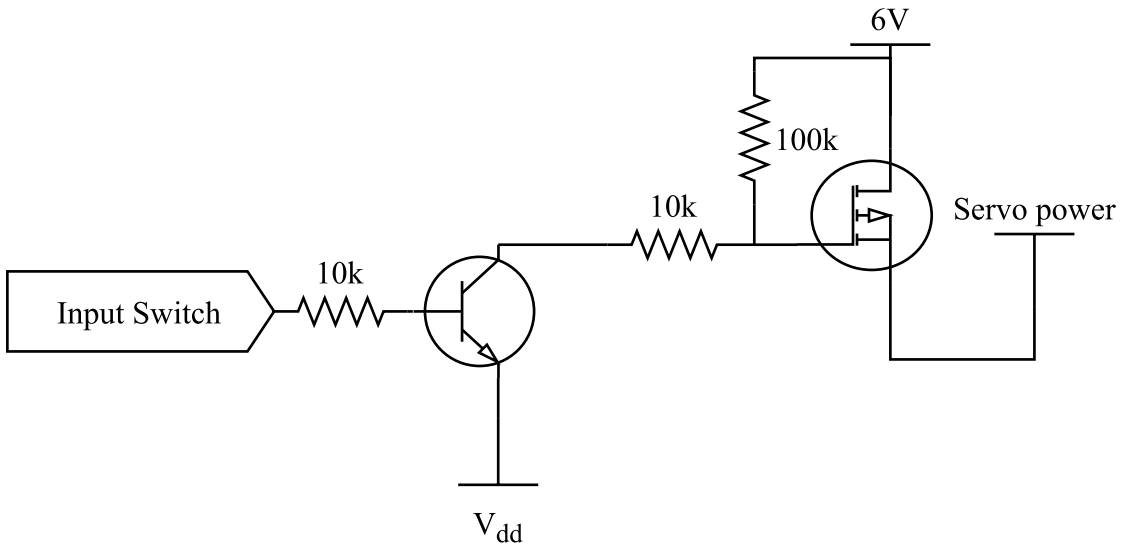

Figure 6. Power switch design.

7. Description of circuit

All of the circuit diagrams displayed above is explained in detail in part 4 of this report

8. Circuit diagrams of modules

The circuit diagrams for all modules are shown in record 6.

9. Description of circuit diagrams

All of the circuit diagrams displayed above is explained in detail in part 4 of this report

10. Timing diagrams

Timing diagrams is not applicable to this project.

11. VHDL code

VHDL code is not applicable to this project.

12. PC board layout

No PCB was designed for this project. The veroboard planning is outlined in record 13

13. Components placement on the board

Since the central part of the robot electronics is the microcontroller used, it is placed in the centre of the Veroboard with the support electronics around it. The DC-DC converter was placed separately from the Veroboard.

14. Wiring diagram of the product mounted in the enclosure

The product has no enclusure.

15. Mechanical design

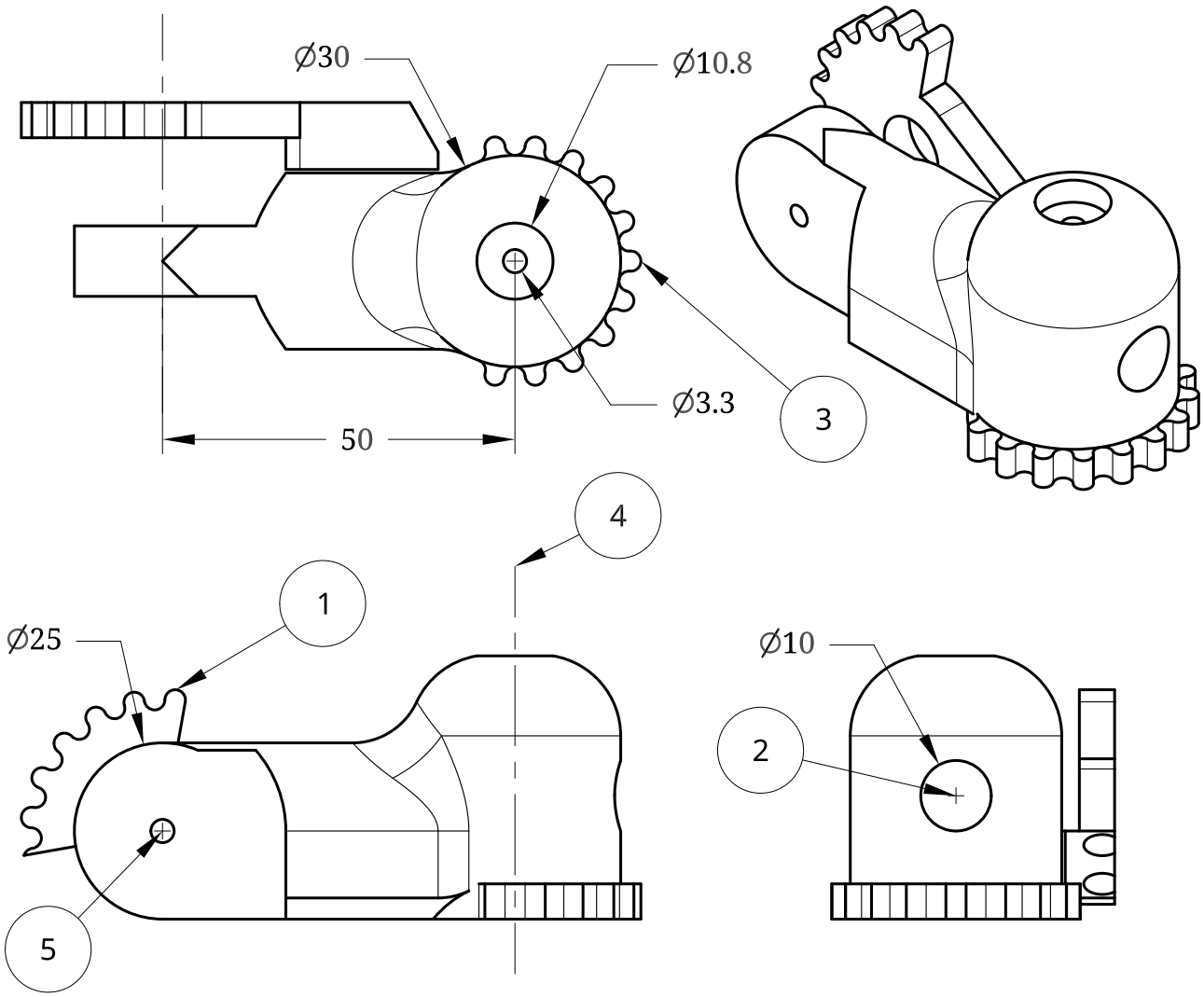

Figure 7. Leg A design.

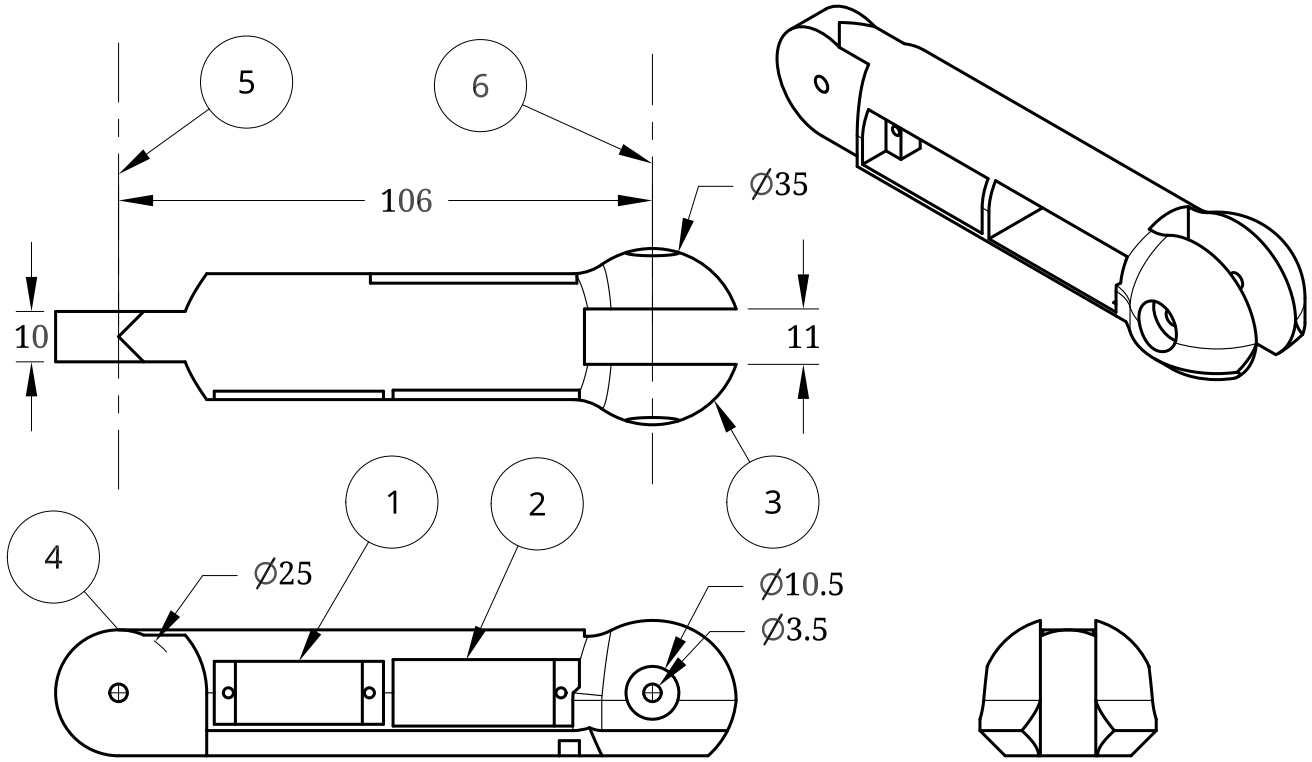

Figure 8. Leg B design.

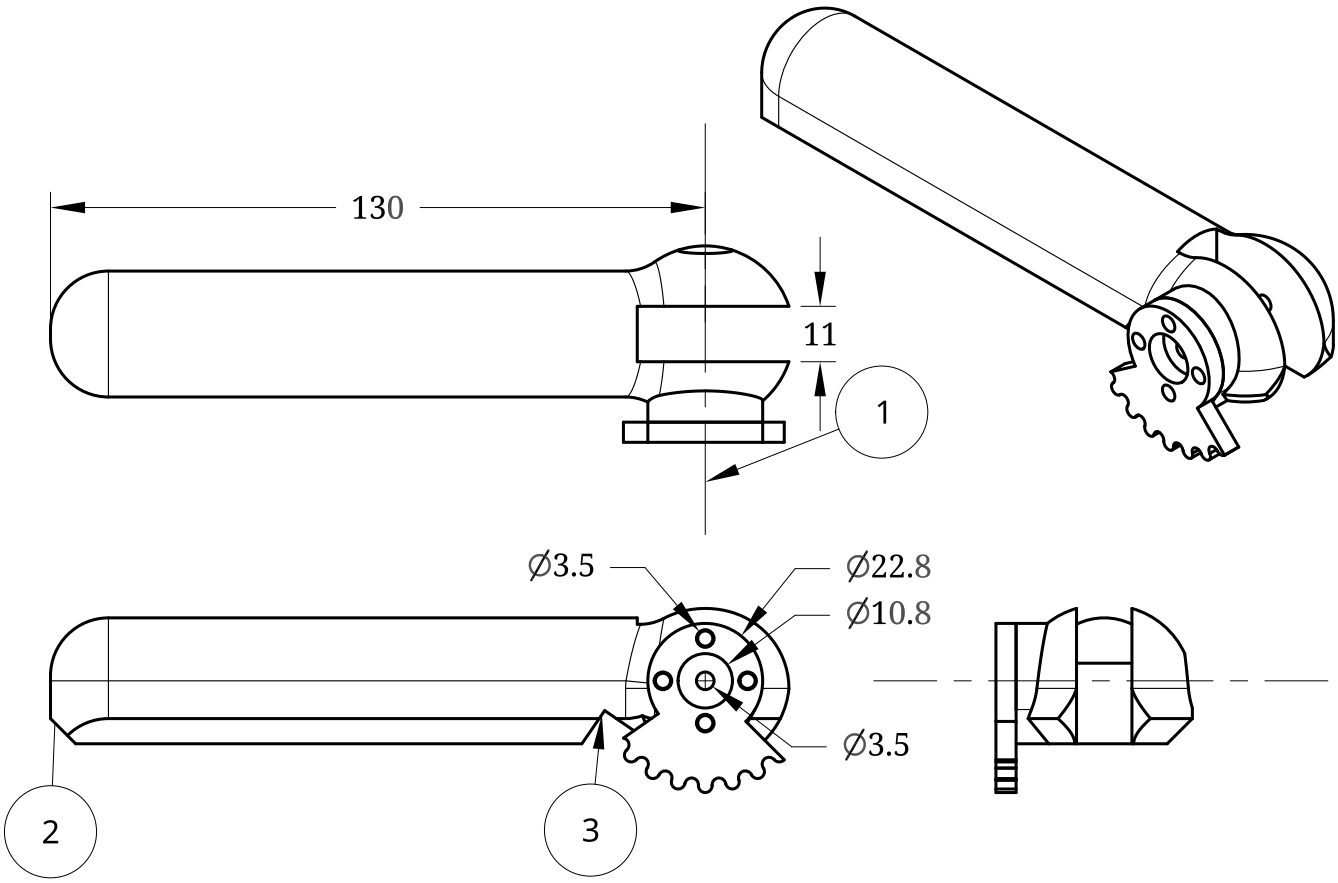

Figure 9. Leg C design.

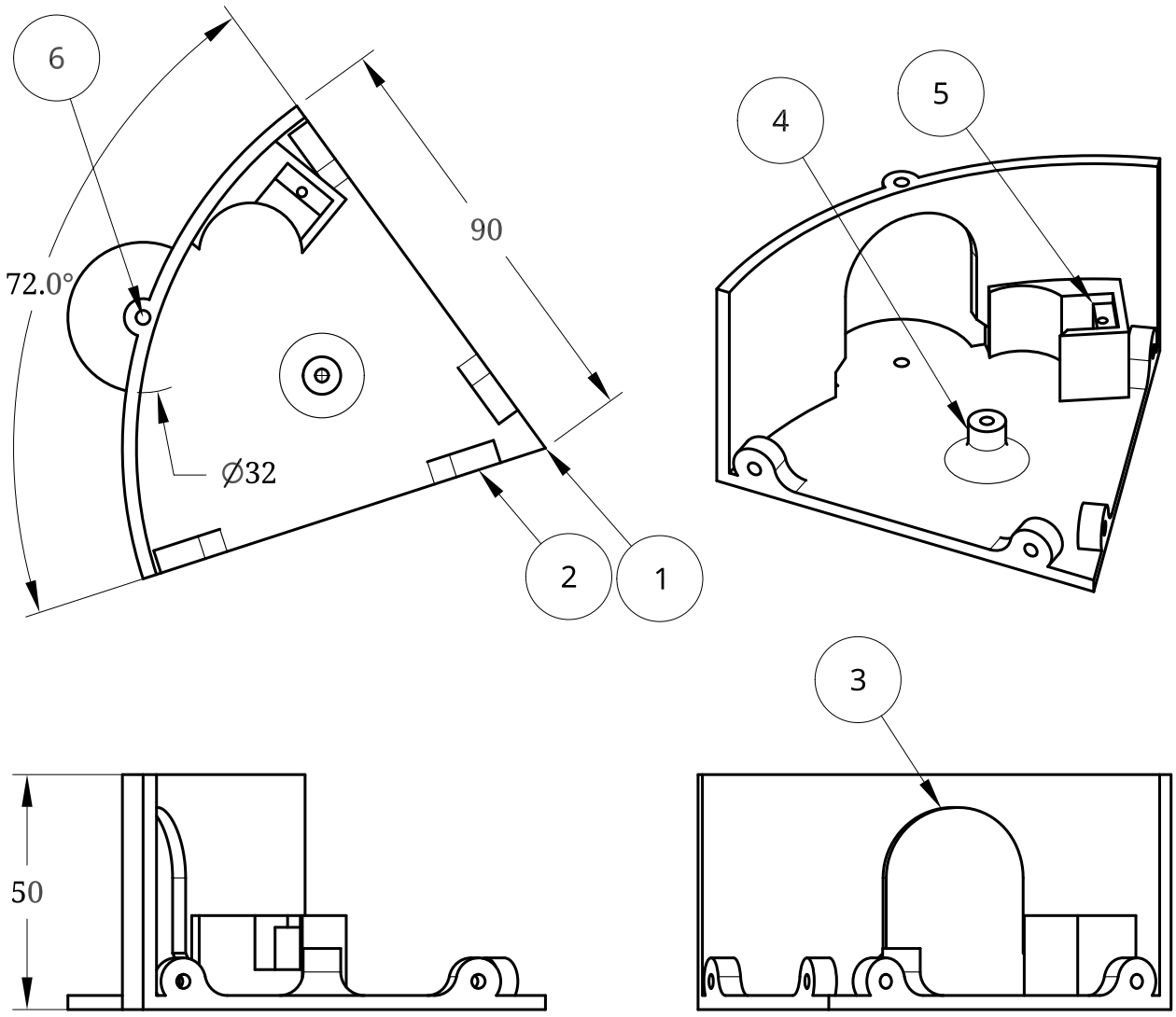

Figure 10. Robot base design.

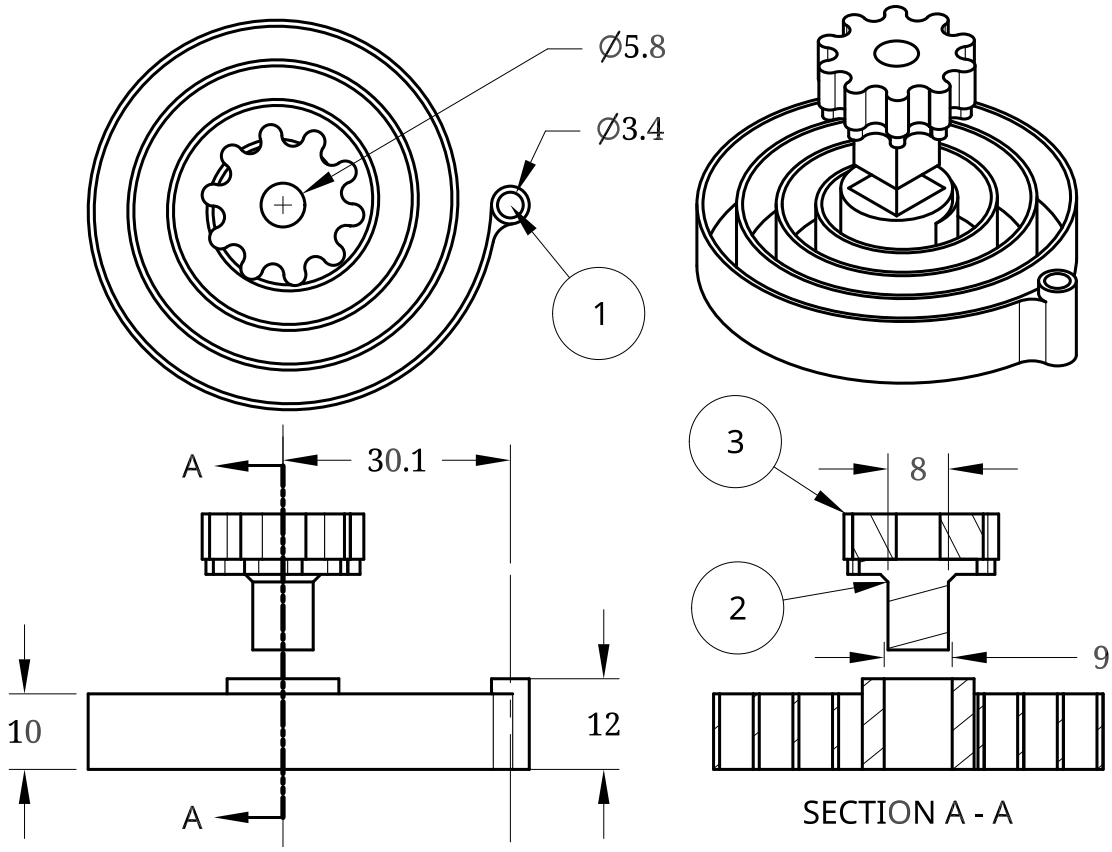

Figure 11. torsional spring design.

16. Acceptance test procedure

Although there is a large software component to the project, the output is purely mechanical. All of the results are therefore pass/fail based on the specification.

17. Pictures of final product

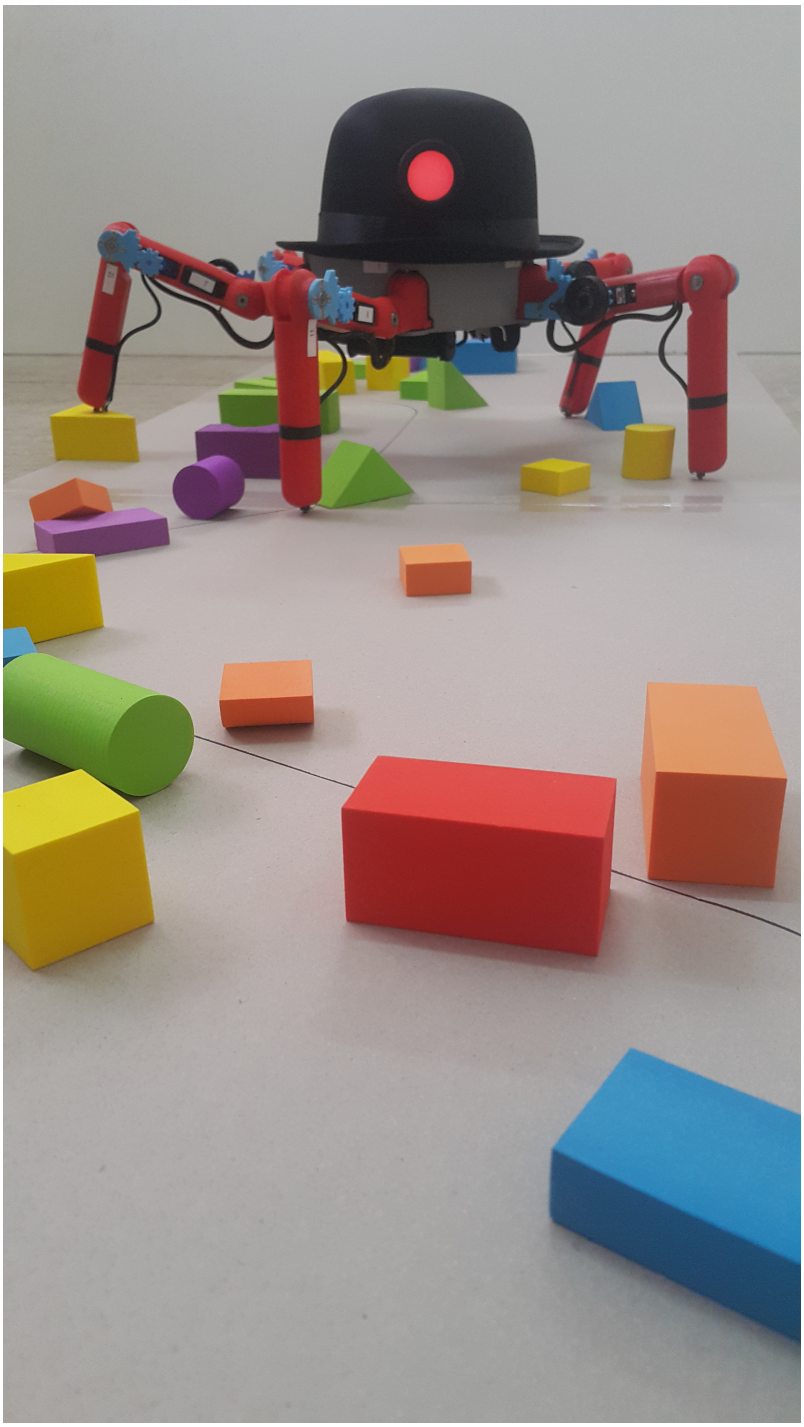

Figure 12. Picture of final product.

18. User guide

The robot is simple to operate. Once the batteries are plugged in, it is ready for Bluetooth connection.

19. List of components cost and suppliers

This was not collected during the course of the project.

20. Description of interfacing with other entities

There are no other entities to be interfaced with.

21. Flow diagram of software

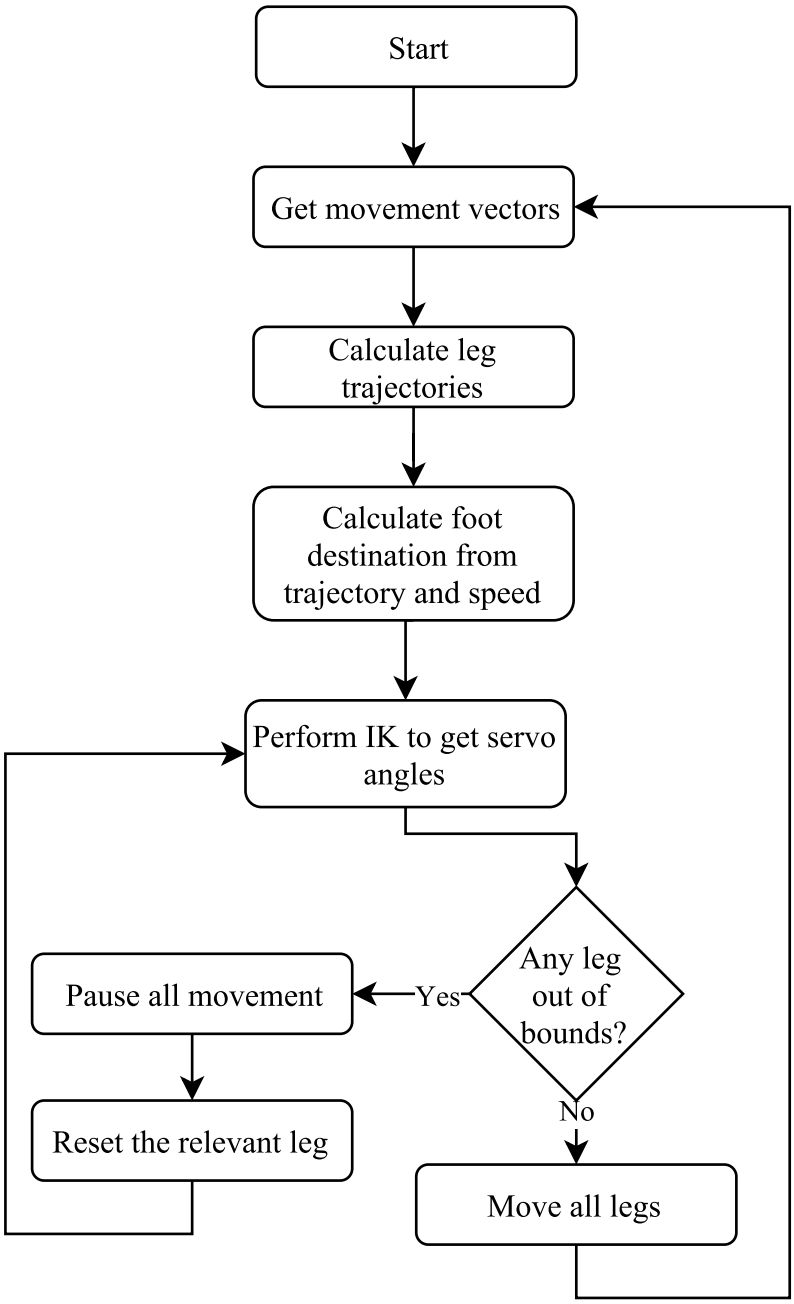

Figure 13. Flow diagram of software.

22. UML diagrams

This is not applicable to this project.

23. Complete source code

```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  ** This notice applies to any and all portions of this file
  * that are not between comment pairs USER CODE BEGIN and
  * USER CODE END. Other portions of this file, whether
  * inserted by the user or by software development tools
  * are owned by their respective copyright owners.
  *
  * COPYRIGHT(c) 2017 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyri
  *      this list of conditions and the following disclaimer in the do
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its co
  *      may be used to endorse or promote products derived from this s
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQ
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE G
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE
  *
  ******************************************************************************
  */
```

```
/* Includes ------------------------------------------------------------------
#include "main.h"
#include "stm32f7xx_hal.h"

/* USER CODE BEGIN Includes */
#include <math.h>
#include <stdbool.h>

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------

I2C_HandleTypeDef hi2c1;

TIM_HandleTypeDef htim6;
TIM_HandleTypeDef htim7;

UART_HandleTypeDef huart1;
UART_HandleTypeDef huart3;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------
unsigned char RXData[15];
unsigned char RXDataBuffer[15];
unsigned char X_Char;
unsigned char Y_Char;
unsigned char R_Char;
int X_Vect =0;
int Y_Vect =0;
int R_Vect =0;
int legAngles[3][15];
int legAngles_id[3][15] = {{1, 2, 3, 4, 5,




int servoOffset[] = {300,315,300,305,295,
```

```
int servoMultiplier[] = {        -360,-380,-363,-365,-370,


int servoCount = 0;
char servoCountChar[2];
int BTCount = 0;
int newPeriod = 0;

static TIM_HandleTypeDef ServoTimer;
static TIM_HandleTypeDef ServoTimer2;

unsigned char BTRewrite[] = {'X','Y','R','\r'};

struct servos {
        int theta;
        int phi;
        int alpha;
};

struct coordinates{
        double x;
        double y;
        double z;
};

struct vectors{
        int X;
        int Y;
        int R;
        bool valid;
};

double rad2deg = 57.29577951308232;
double A_length = 50;
double B_length = 106;
double C_length = 130;
double z_body = 70;
double robotRadius = 90;

bool resetStatus[5];

double X_Vector = 0;
double Y_Vector = 0;
double R_Vector = 0;
```

```
double destination[2][5];

double currentPosition[2][5];

double translateLeg[2][5];

bool legAngleFlag = false;
int bufferLevel;
int time = 0;
bool BTon = false;
//Peripheral pin & port difinitions
#define BTPort                                  GPIOE
#define BTLEDPin                                GPIO_PIN_0
#define redLEDPort                      GPIOB
#define redLEDPin                       GPIO_PIN_9
#define greenLEDPort            GPIOB
#define greenLEDPin             GPIO_PIN_8
#define servoPowerPort  GPIOC
#define servoPowerPin           GPIO_PIN_9

//Servo pin & port definitions
#define servo1Pin               GPIO_PIN_12
#define servo1Port              GPIOB
#define servo2Pin               GPIO_PIN_13
#define servo2Port              GPIOB
#define servo3Pin               GPIO_PIN_14
#define servo3Port              GPIOB
#define servo4Pin               GPIO_PIN_15
#define servo4Port              GPIOB
#define servo5Pin               GPIO_PIN_8
#define servo5Port              GPIOD
#define servo6Pin               GPIO_PIN_9
#define servo6Port              GPIOD
#define servo7Pin               GPIO_PIN_10
#define servo7Port              GPIOD
#define servo8Pin               GPIO_PIN_11
#define servo8Port              GPIOD
#define servo9Pin               GPIO_PIN_12
#define servo9Port              GPIOD
#define servo10Pin              GPIO_PIN_13
#define servo10Port     GPIOD
#define servo11Pin              GPIO_PIN_14
#define servo11Port     GPIOD
```

```
#define servo12Pin                GPIO_PIN_15
#define servo12Port      GPIOD
#define servo13Pin                GPIO_PIN_6
#define servo13Port      GPIOC
#define servo14Pin                GPIO_PIN_7
#define servo14Port      GPIOC
#define servo15Pin                GPIO_PIN_8
#define servo15Port      GPIOC

uint16_t servoPinArray[] ={      servo1Pin, servo2Pin, servo3Pin,



GPIO_TypeDef * servoPortArray[] = {      servo1Port, servo2Port, servo3P



/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_TIM6_Init(void);
static void MX_TIM7_Init(void);
static void MX_I2C1_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------
void Debug(char *Array, int count);
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart);
void SetupTimers(void);
void Sort(void);
void SetServo(int servo, int degrees);
struct servos IK( struct coordinates input);
bool CheckBounds(struct servos leg);
struct coordinates Rotate(double x, double y, double angle);     //Rotat
void TranslateLeg(void);
struct coordinates Vector(double x, double y, int leg, bool offset, dou
void StartPosition(void);
```

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
void BTDecrypt(void);
void ServoUpdate(void);
void Delay(int a);
void ResetLeg(int leg);
void DestinationUpdate(int leg);

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* 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
  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_USART3_UART_Init();
  MX_USART1_UART_Init();
  MX_TIM6_Init();
  MX_TIM7_Init();
  MX_I2C1_Init();
```

```
/* USER CODE BEGIN 2 */
        SetupTimers();
        StartPosition();

        TranslateLeg();

        Debug("Init done",9);

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
        Debug("Entering main loop",18);
        HAL_UART_Receive_IT(&huart3,RXDataBuffer,15);
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
                    //Turn off all LEDs except red
                    HAL_GPIO_WritePin(BTPort,BTLEDPin,GPIO_PIN_SET);
                    HAL_GPIO_WritePin(greenLEDPort,greenLEDPin,GPIO_PIN_SET
                    HAL_GPIO_WritePin(redLEDPort,redLEDPin,GPIO_PIN_RESET);

                    Debug("Waiting for Bluetooth...",24);

                    while (BTon == false)
                    {
                            //Wait here for connection
                    }
                    //Turn off all LEDs except blue
                    HAL_GPIO_WritePin(BTPort,BTLEDPin,GPIO_PIN_RESET);
                    HAL_GPIO_WritePin(greenLEDPort,greenLEDPin,GPIO_PIN_SET
                    HAL_GPIO_WritePin(redLEDPort,redLEDPin,GPIO_PIN_SET);
                    Debug("Turning on servos...",20);
                    //Move servos to home position
                    StartPosition();
                    //Turn on power to servos
                    HAL_GPIO_WritePin(servoPowerPort,servoPowerPin,GPIO_PIN
                    //Delay for 1 second to get things started
                    Delay(10);
                    //Put legs in the 0,0,50 position one at a time
                    for (int leg = 1; leg <6; leg++)
                    {
```

```
                        struct coordinates leg_Vector = Vector(0,0,leg,
                        leg_Vector.z = 50;                          //Lift
                        struct servos leg_servo = IK(leg_Vector);

                        SetServo(leg,leg_servo.theta);
                        SetServo(leg+5,leg_servo.phi);
                        SetServo(leg+10,leg_servo.alpha);
                        ServoUpdate();
                        Delay(10);
                }
                //Slowly raise the body
                for (int height = 50 ; height >0 ; --height)
                {
                        for (int leg = 1; leg <6; leg++)
                        {
                                struct coordinates leg_Vector = Vector(
                                leg_Vector.z = height;
                                struct servos leg_servo = IK(leg_Vector

                                SetServo(leg,leg_servo.theta);
                                SetServo(leg+5,leg_servo.phi);
                                SetServo(leg+10,leg_servo.alpha);
                                ServoUpdate();
                        }
                        Delay(2);
                }
                //Turn off all LEDs except green
                HAL_GPIO_WritePin(BTPort,BTLEDPin,GPIO_PIN_SET);
                HAL_GPIO_WritePin(greenLEDPort,greenLEDPin,GPIO_PIN_RESE
                HAL_GPIO_WritePin(redLEDPort,redLEDPin,GPIO_PIN_SET);
                Debug("Ready...",8);

                //This is the main part that executes while the bluetoo
                while(BTon)
                {
                        //Get the vectors from the last transmitted mes
                        BTDecrypt();
                        //Perform inverse kinematics
                        for (int leg = 1; leg <6; leg++)
                        {
//                                      struct coordinates leg_Vector = Vector(-
//!!!!!!!!!This does not accumulate!!!!!!!!!!!!!!!!!!!!
                                DestinationUpdate(leg);
                                struct coordinates leg_Vector;
```

```
                        leg_Vector.x = currentPosition[0][leg-1
                        leg_Vector.y = currentPosition[1][leg-1
                        leg_Vector.z = 0;
                        struct servos leg_servo = IK(leg_Vector
                        //Check if this is out ouf bounds
                        if (CheckBounds(leg_servo))
                        {
                                resetStatus[leg-1] = true;
                        }else{
                                resetStatus[leg-1] = false;
                        }
                        SetServo(leg,leg_servo.theta);
                        SetServo(leg+5,leg_servo.phi);
                        SetServo(leg+10,leg_servo.alpha);
                }
                //Reset legs if necessary
                for (int leg = 1; leg < 6; leg++)
                {
                        if (resetStatus[leg-1] == true)
                        {
                                //Turn on all LEDs (white)
                                HAL_GPIO_WritePin(BTPort,BTLEDP
                                HAL_GPIO_WritePin(greenLEDPort,
                                HAL_GPIO_WritePin(redLEDPort,red
                                //Reset leg
                                ResetLeg(leg);
                        }
                }
                //Move the legs to where we want them
                ServoUpdate();
                //Turn off all LEDs except green
                HAL_GPIO_WritePin(BTPort,BTLEDPin,GPIO_PIN_SET)
                HAL_GPIO_WritePin(greenLEDPort,greenLEDPin,GPIO
                HAL_GPIO_WritePin(redLEDPort,redLEDPin,GPIO_PIN
        }


  }
  /* USER CODE END 3 */

}

/** System Clock Configuration
```

```
*/
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 10;
  RCC_OscInitStruct.PLL.PLLN = 216;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Activate the Over-Drive mode
    */
  if (HAL_PWREx_EnableOverDrive() != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**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_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
```

```
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_7) != HAL_O
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_P
                                |RCC_PERIPHCLK_I2C1;
  PeriphClkInitStruct.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
  PeriphClkInitStruct.Usart3ClockSelection = RCC_USART3CLKSOURCE_PCLK1;
  PeriphClkInitStruct.I2c1ClockSelection = RCC_I2C1CLKSOURCE_PCLK1;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**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);
}

/* I2C1 init function */
static void MX_I2C1_Init(void)
{

  hi2c1.Instance = I2C1;
  hi2c1.Init.Timing = 0x20404768;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
```

```
  }

    /**Configure Analogue filter
    */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) !=
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure Digital filter
    */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/* TIM6 init function */
static void MX_TIM6_Init(void)
{

  TIM_MasterConfigTypeDef sMasterConfig;

  htim6.Instance = TIM6;
  htim6.Init.Prescaler = 0;
  htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim6.Init.Period = 0;
  htim6.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim6, &sMasterConfig) !=
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/* TIM7 init function */
```

```
static void MX_TIM7_Init(void)
{

  TIM_MasterConfigTypeDef sMasterConfig;

  htim7.Instance = TIM7;
  htim7.Init.Prescaler = 0;
  htim7.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim7.Init.Period = 0;
  htim7.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim7) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim7, &sMasterConfig) !=
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/* 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;
  huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
```

```
/* USART3 init function */
static void MX_USART3_UART_Init(void)
{

  huart3.Instance = USART3;
  huart3.Init.BaudRate = 9600;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/** 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_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOE_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8
```

```
                          |GPIO_PIN_9, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_
                          |GPIO_PIN_15|GPIO_PIN_8|GPIO_PIN_9, GPIO_PIN_

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_1
                          |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN
                          |GPIO_PIN_1, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_0|GPIO_PIN_1, GPIO_PIN_RESET);

  /*Configure GPIO pins : PC1 PC6 PC7 PC8
                           PC9 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8
                          |GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PA1 */
  GPIO_InitStruct.Pin = GPIO_PIN_1;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PA4 PA5 PA6 PA7 */
  GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PC4 */
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
  /*Configure GPIO pins : PB1 PB12 PB13 PB14
                          PB15 PB8 PB9 */
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14
                        |GPIO_PIN_15|GPIO_PIN_8|GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pins : PD8 PD9 PD10 PD11
                          PD12 PD13 PD14 PD15
                          PD1 */
  GPIO_InitStruct.Pin = GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11
                        |GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN
                        |GPIO_PIN_1;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

  /*Configure GPIO pins : PE0 PE1 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/**
  * @brief  This function sends a string to UART1 to be received in a t
        * @param  *Array is a pointer to the string
                                                  count is an int equal t
  * @retval None
  */
void Debug(char *Array, int count)
{
        uint8_t TXBuffer[count+1];
        for(int i = 0; i!=count;i++)
        {
                TXBuffer[i] = Array[i];
```

```
        }

        TXBuffer[count] ='\r';
        TXBuffer[count+1] ='\n';
        HAL_UART_Transmit(&huart1 ,TXBuffer ,count+2 ,10);
}

/**
  * @brief  This function is executed in the case of an error with the
        * @param  *huart is a handle of the UART module in question
  * @retval None
  */
void HAL_UART_ErrorCallback(UART_HandleTypeDef *huart)
{
        if (huart == &huart3)
        {
                Debug("Bluetooth␣Error",15);
        }else if (huart == &huart1)
        {
                Debug("Serial␣Error",12);
        }else{
                Debug("UART␣error",10);
        }

}

/**
  * @brief  This function configures the timers used in generating the
        * @param  None
  * @retval None
  */
void SetupTimers(void)
{
        //Timer 6
        //50Hz Interrupt
        ServoTimer.Instance =TIM6;
        ServoTimer.Init.Prescaler =40000;
        ServoTimer.Init.CounterMode = TIM_COUNTERMODE_UP;
        ServoTimer.Init.Period =54;
        ServoTimer.Init.RepetitionCounter = 0;
        HAL_TIM_Base_Init(&ServoTimer);
        HAL_TIM_Base_Start_IT(&ServoTimer);

        //Timer 7
```

```
        //Variable frequency interrupt
        htim7.Instance =TIM7;
        htim7.Init.Prescaler =1000;
        htim7.Init.CounterMode = TIM_COUNTERMODE_UP;
        htim7.Init.Period =160;
        htim7.Init.RepetitionCounter = 0;
        HAL_TIM_Base_Init(&htim7);
        HAL_TIM_Base_Start_IT(&htim7);
}


/** @brief Function for sorting the desired leg angles into ascending o
                                                    to make timer functions
  * @param None
  * @retval None */
void Sort(void)
{
        for(int i = 0;i<15;i++)
        {
                for(int j = i+1;j<15;j++)
                {
                        if(legAngles[0][i] > legAngles[0][j])
                        {
                                //swop the angles
                                int a = legAngles[0][i];
                                legAngles[0][i] = legAngles[0][j];
                                legAngles[0][j] = a;
                                //now swop the index
                                a = legAngles_id[0][i];
                                legAngles_id[0][i] = legAngles_id[0][j]
                                legAngles_id[0][j] = a;
                        }
                }
        }
}

/**
  * @brief  This function executes when a timer interrupts. It is used
        * @param  *htim is a handle of the timer module that triggered
  * @retval None
  */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
        {
```

```
if(htim == &htim7){
        if(legAngleFlag == true)
        {
                bufferLevel = 2;
        }else{
                bufferLevel = 1;
        }
        //Turn off
        ++servoCount;
        if (servoCount < 15)
        {
                newPeriod = (legAngles[bufferLevel][serv

                while (newPeriod == 0)
                {
                        HAL_GPIO_WritePin(servoPortArray
                        ++servoCount;
                        newPeriod = (legAngles[bufferLe
                }
                //Adjust for time lost in this subrouti
                if (newPeriod>1)
                {
                        --newPeriod;
                }
                if (newPeriod < 0 || newPeriod > 400)
                {
                        newPeriod = 0;
                }
                TIM7->ARR = (newPeriod);
                HAL_GPIO_WritePin(servoPortArray[legAng
        }else{
                //Turn off servo 15
                HAL_GPIO_WritePin(servoPortArray[legAng
                //Stop the timer for the rest of the cy
                HAL_TIM_Base_Stop(&htim7);
                if (servoCount != 0xF)
                {
                        Debug("servoCount Error",16);
                }
                servoCount = 0;
        }
}else if (htim == &htim6)
{
        servoCount = 0;
```

```
++time;
//50Hz Interrupt
//Turn all on
HAL_GPIO_WritePin(servo1Port,servo1Pin,GPIO_PIN
HAL_GPIO_WritePin(servo2Port,servo2Pin,GPIO_PIN
HAL_GPIO_WritePin(servo3Port,servo3Pin,GPIO_PIN
HAL_GPIO_WritePin(servo4Port,servo4Pin,GPIO_PIN
HAL_GPIO_WritePin(servo5Port,servo5Pin,GPIO_PIN
HAL_GPIO_WritePin(servo6Port,servo6Pin,GPIO_PIN
HAL_GPIO_WritePin(servo7Port,servo7Pin,GPIO_PIN
HAL_GPIO_WritePin(servo8Port,servo8Pin,GPIO_PIN
HAL_GPIO_WritePin(servo9Port,servo9Pin,GPIO_PIN
HAL_GPIO_WritePin(servo10Port,servo10Pin,GPIO_P
HAL_GPIO_WritePin(servo11Port,servo11Pin,GPIO_P
HAL_GPIO_WritePin(servo12Port,servo12Pin,GPIO_P
HAL_GPIO_WritePin(servo13Port,servo13Pin,GPIO_P
HAL_GPIO_WritePin(servo14Port,servo14Pin,GPIO_P
HAL_GPIO_WritePin(servo15Port,servo15Pin,GPIO_P


if(legAngleFlag == true)
{
        bufferLevel = 2;
}else{
        bufferLevel = 1;
}
//Set period for 1st servo
newPeriod = legAngles[bufferLevel][servoCount];
if (newPeriod < 0 || newPeriod > 400)
        {
                newPeriod = 0;
        }
while (newPeriod == 0 && servoCount < 16)
{
        ++servoCount;
        newPeriod = legAngles[bufferLevel][servo
        if (newPeriod < 0 || newPeriod > 400)
        {
                newPeriod = 0;
        }
}


TIM7->ARR = newPeriod;
```

```
                        //Start the timer
                        HAL_TIM_Base_Start(&htim7);

                        //BT LED
                        ++BTCount;
                        if (BTCount > 120) //20ms * 120 = 2.4s
                        {
                                //Assume BT is disconnected
                                BTon = false;
                                //HAL_GPIO_WritePin(servoPowerPort,servo
                                BTCount = 0;
//                              HAL_GPIO_TogglePin(greenLEDPort,greenLE
//                              HAL_GPIO_TogglePin(redLEDPort,redLEDPin
                        }
                }else{
                        Debug("Timer Interrupt Error",21);
                }
        }

/**
  * @brief  This function calculates the required timer values for each
        * @param  servo is an int in the range 1:15 corresponding to a
                                                degrees is an int in th
  * @retval None
  */
void SetServo(int servo, int degrees)
{
        //Set the servo with the correct index in the array
        for (int i = 0; i<15; i++)
        {
                if (servo == legAngles_id[0][i])
                {
                        legAngles[0][i] =servoOffset[servo-1] + (degree
                        return;
                }
        }
}

/**
  * @brief  This function does the inverse kinematic calculations for a
        * @param  input is a struct of type coordinates with members x,
                                                corresponding t
  * @retval servos is a struct with members theta, phi and alpha corresp
                                                resulting angle
```

```
  */
struct servos IK( struct coordinates input)
{
        //make the return struct
        struct servos result;
        //Calculate theta
        double theta = atan2(input.x,input.y);
        //Calculate coordinates of joint 2
        double x = A_length * sin(theta);
        double y = A_length * cos(theta);
        //horizontal distance from joint 1 to foot
        double reach = sqrt(pow(input.x-x,2)+pow(input.y-y,2));
        //Absolute distance between joint 2 and foot
        double twoToFoot = sqrt(pow(input.x-x,2)+pow(input.y-y,2) +pow(
        //perform Cosine rule to get alpha
        double f =((pow(B_length ,2) + pow(C_length,2) - pow(twoToFoot
        double d = acos(f);
        double alpha =(rad2deg * d) + 0.5;                //+0.5 for roun
        result.alpha = (int)alpha;
        //use sine rule to get phi
        double c = asin(C_length * sin(d) / twoToFoot);
        double e = atan2(reach ,z_body-input.z);
        double phi = 270 - c*rad2deg - e*rad2deg + 0.5; // +0.5 for rou
        result.phi = (int)phi;
        //convert theta to degrees. Add 0.5 for rounding
        theta = (theta * rad2deg) + 0.5;
        result.theta = 180-(int)theta;

        return result;
}

/**
  * @brief  This function checks that all legs are within the boundarie
                                                        writes the resu
        * @param  None
  * @retval None
  */
bool CheckBounds(struct servos leg)
{
        if ( leg.theta < 50 || leg.theta > 120          // 50 < theta <
                || leg.phi < 90 || leg.phi > 170                // 90 <
< 170
                || leg.alpha < 70 || leg.alpha> 130)            // 70 <
        {
```

```
                return true;
        }else{
                return false;
        }
}

/**
  * @brief  This function rotates the given coordinates about the origi
        * @param  x,y are doubles corresponding to the coordinates
                                                angle is a double in de
  * @retval a coordinates struct with members x, y and z. z is not used
  */
struct coordinates Rotate(double x, double y, double angle)
{
        //Create output struct
        struct coordinates result;
        //convert angle to radians
        angle = angle/rad2deg;
        //calculate return values
        result.x = x * cos(angle) - y * sin(angle);
        result.y = x * sin(angle) + y * cos(angle);

        return result;
}

/**
  * @brief  This function calculates the normalized position of a leg fr
                                                        given movement
        * @param          x is a double with the X vector
                                                y is a double with the
                                                leg is an int in the ra
                                                offset is a bool indica
                                                        denormalized
  * @retval a struct of type coordinates. The z member is unused
  */
struct coordinates Vector(double x, double y, int leg, bool offset, dou
{
        //Create return struct
        struct coordinates result;
        //Make the trajectory polar with a magnitude and angle
        double trajectory = atan2(y,x);
        double magnitude = sqrt(pow(x,2)+pow(y,2));
        //Match the trajectory to the leg
        struct coordinates neutral;
```

```
        //Translation components
        if (offset)
        {
                neutral = Rotate(270,0,(leg-1)*72);
                struct coordinates offset= Rotate(robotRadius,0,(leg-1)
                neutral.x = neutral.x - offset.x;
                neutral.y = neutral.y - offset.y;
        }else{
                neutral.x = 0;
                neutral.y = 0;
        }
        result.x = neutral.x + magnitude*cos(trajectory) + translateLeg
        result.y = neutral.y + magnitude*sin(trajectory) + translateLeg
        //+ Rotation
        struct coordinates temp = Rotate(result.x,result.y,-(leg-1)*72-
        result.x = temp.x - translateLeg[0][0];
        result.y = temp.y - translateLeg[1][0];
        if (offset)
        {
                //save the destination for
use in reset leg function later.
                destination[0][leg-1] = result.x;
                destination[1][leg-1] = result.y;
        }
        return result;
}

/**
  * @brief  This function is used once to calculate coefficients and st
        * @param  None
  * @retval None
  */
void TranslateLeg(void)
{
        for(int leg = 0; leg < 5; leg++)
        {
                translateLeg[0][leg] = robotRadius * cos(leg*72/rad2deg
                translateLeg[1][leg] = robotRadius * sin(leg*72/rad2deg
        }
}

/**
  * @brief  This function resets a leg if it is out of bounds
        * @param  leg is an int in the range 1:5
```

```
  * @retval None
  */
void ResetLeg(int leg)
{
        //Lift up the leg where it is currently
        struct coordinates lift;
        lift.x = destination[0][leg-1];          //destination array is
        lift.y = destination[1][leg-1];
        lift.z = 30;
        struct servos leg_up = IK(lift);
        //Send this to servos and wait
        SetServo(leg,leg_up.theta);
        SetServo(leg+5,leg_up.phi);
        SetServo(leg+10,leg_up.alpha);
        ServoUpdate();
        Delay(10);
        //Now move the leg over to the new position
        struct coordinates reset = Vector(X_Vect,Y_Vect,leg,true,R_Vect
        reset.z = 30;
        destination[0][leg-1] = reset.x;
        destination[1][leg-1] = reset.y;
        currentPosition[0][leg-1] = reset.x;
        currentPosition[1][leg-1] = reset.y;
        struct servos leg_reset = IK(reset);
        //Send this to servos and wait
        SetServo(leg,leg_reset.theta);
        SetServo(leg+5,leg_reset.phi);
        SetServo(leg+10,leg_reset.alpha);
        ServoUpdate();
        Delay(10);
        //Now put down the leg where it is
        reset.z = 0;
        leg_reset = IK(reset);
        //Send this to the servos and wait
        SetServo(leg,leg_reset.theta);
        SetServo(leg+5,leg_reset.phi);
        SetServo(leg+10,leg_reset.alpha);
        ServoUpdate();
        Delay(10);
}

void StartPosition(void)
{
        //Set servo begin positions
```

```
        SetServo(1,90);
        SetServo(2,90);
        SetServo(3,90);
        SetServo(4,90);
        SetServo(5,90);
        SetServo(6,100);
        SetServo(7,100);
        SetServo(8,100);
        SetServo(9,100);
        SetServo(10,100);
        SetServo(11,90);
        SetServo(12,90);
        SetServo(13,90);
        SetServo(14,90);
        SetServo(15,90);
        ServoUpdate();
        Delay(10);
}
/**
  * @brief  This is an Interrupt service routine for when bluetooth com
  * @param  None
  * @retval None
  */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
        BTon = true;
        for (int i = 0;i<16;i++)
        {
                RXData[i] = RXDataBuffer[i];
        }
        HAL_UART_Receive_IT(&huart3,RXDataBuffer,15);
}

void BTDecrypt(void)
{
        //Bluetooth reception decryption
                for(int i = 0;i<7;i++)
                {
                        //Is this a valid message?
                        if ((RXData[i] == 'X')&& (RXData[i+2] == 'Y')&&
                        {
//                              //Turn Blue LED on
//                              HAL_GPIO_WritePin(BTPort,BTLEDPin,GPIO_
//                              HAL_GPIO_WritePin(servoPowerPort,servoP
```

```
                                        BTon = true;
                                        BTCount = 0;
                                        //Extract the necessary commands
                                        X_Char = RXData[i+1];
                                        Y_Char = RXData[i+3];
                                        R_Char = RXData[i+5];
                                        //Clear the array
                                        for(int j = 0;j<14;j++)
                                        {
                                                RXData[j]=0;
                                        }
                                        //Calculate correct vectors
                                        X_Vect = X_Char - '5';
                                        Y_Vect = Y_Char - '5';
                                        R_Vect = R_Char - '5';
                                        //Debug
                                        HAL_UART_Transmit(&huart1,&X_Char,1,10)
                                        HAL_UART_Transmit(&huart1,&Y_Char,1,10)
                                        HAL_UART_Transmit(&huart1,&R_Char,1,10)
                                        HAL_UART_Transmit(&huart1,BTRewrite,4,1
                                        }
                        }
}

void ServoUpdate(void)
{
        //Set a flag while editing these registers
        legAngleFlag = true;
        //Sort the servo times in ascending order
        Sort();
        //Copy the results into buffer level 2
                for (int n = 0; n<15 ;n++)
        {
        legAngles[1][n] = legAngles[0][n];
        legAngles_id[1][n] = legAngles_id[0][n];
        }
        //Clear busy flag
        legAngleFlag = false;
        //Copy into buffer level 3
        for (int n = 0; n<15 ;n++)
        {
        legAngles[2][n] = legAngles[1][n];
        legAngles_id[2][n] = legAngles_id[1][n];
        }
```

```
}

void Delay(int a)
{
        time = 0;
        while(time < a)
        {
                //wait here till delay value is reached.
        }
}

void DestinationUpdate(int leg)
{
        //Move destination to current
        currentPosition[0][leg-1] = destination[0][leg-1];
        currentPosition[1][leg-1] = destination[1][leg-1];
        //Vector to add to this
//      struct coordinates leg_Vector = Vector(-X_Vect,-Y_Vect,leg,true
//      leg_Vector.z = 0;
        struct coordinates newVector = Vector(-X_Vect,-Y_Vect,leg,false
        //Destination = current + new
        destination[0][leg-1] = newVector.x + currentPosition[0][leg-1]
        destination[1][leg-1] = newVector.y + currentPosition[1][leg-1]
}
/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  None
  * @retval None
  */
void _Error_Handler(char * file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return
  while(1)
  {
  }


  /* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
```

```
/**
    * @brief Reports the name of the source file and the source line num
    * 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 li
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
  /* USER CODE END 6 */

}

#endif

/**
  * @}
  */

/**
  * @}
*/

/************************ (C) COPYRIGHT STMicroelectronics *****END OF
```

24. Explanation of software modules

All of the software modules are explained in part 4 of this report.

25. Simulations

```
##############################################################
```

```
#
#
# Author: L Steyn
#
#
#
# Code from Inverse Kinematics merged with UI test code
#
# Implemented in different threads to enable live plotting
#
#
#
#################################################################

import sys
from PyQt4 import QtGui
from PyQt4 import QtCore
from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as Fig
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.art3d as art3d
import math
from matplotlib.patches import Circle
# Predefine Vectors
X_Vector = 0
Y_Vector = 0
Rot_Vector = 0

# Predefine variables
Z_body = 0.5
# Height of the robot body
A = 0.5
# Leg segment A length
B = 1
# Leg segment B length
C = 1
# Leg segment C length
Radius = 1
# Radius of robot body
Z = 0
Time = 2
# This is the update frequency of the system in seconds
Speed = 0.1
# This is the speed of the robot (normalized)
destination = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
# Init to zeros
current = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
# Init to zeros
StepSize = 0
reset =[False,False,False,False,False]

# Create GUI class with all the components necessary to create and main
class GUI(QtGui.QDialog):

    def __init__(self, parent=None):
# Main function of the GUI class. Inits everything
        super(GUI, self).__init__(parent)
        self.figure = plt.figure()
# Create instance of matplotlib figure for GUI
        GUI.canvas = FigureCanvas(self.figure)
# Put this figure on a canvas object
        plt.ion()

        # Create UI widgets
        self.PlotButton = QtGui.QPushButton('Step')
# Plot button
        self.PlotButton.setFixedSize(50, 50)
        self.PlotButton.clicked.connect(self.Step)
        self.UpButton = QtGui.QPushButton('UP')
# Forward button
        self.UpButton.setFixedSize(50, 50)
        self.UpButton.clicked.connect(self.UP)
        self.DownButton = QtGui.QPushButton('DOWN')
# Reverse button
        self.DownButton.setFixedSize(50, 50)
        self.DownButton.clicked.connect(self.DOWN)
        self.LeftButton = QtGui.QPushButton('LEFT')
# Left button
        self.LeftButton.setFixedSize(50, 50)
        self.LeftButton.clicked.connect(self.LEFT)
        self.RightButton = QtGui.QPushButton('RIGHT')
# Right button
        self.RightButton.setFixedSize(50, 50)
        self.RightButton.clicked.connect(self.RIGHT)
        self.CWButton = QtGui.QPushButton('CW')
# Rotate clockwise button
        self.CWButton.setFixedSize(50, 50)
        self.CWButton.clicked.connect(self.CW)
        self.CCWButton = QtGui.QPushButton('CCW')
```

```
# Rotate counterclockwise button
        self.CCWButton.setFixedSize(50, 50)
        self.CCWButton.clicked.connect(self.CCW)
        self.D11 = QtGui.QLabel('X Vector:')
# X vector text label
        self.D11.setFixedWidth(75)
        self.D12 = QtGui.QLabel('0')
# X vector numerical label
        self.D12.setFixedWidth(75)
        self.D21 = QtGui.QLabel('Y Vector:')
# Y vector text label
        self.D21.setFixedWidth(75)
        self.D22 = QtGui.QLabel('0')
# Y vector numerical label
        self.D22.setFixedWidth(75)
        self.D31 = QtGui.QLabel('Rotation:')
# Rotation text label
        self.D31.setFixedWidth(75)
        self.D32 = QtGui.QLabel('0')
# Rotation numerical label
        self.D32.setFixedWidth(75)
        self.AutoPlot = QtGui.QCheckBox('Auto Plot')
# Checkbox to enable plotting on press of a a directional button

        # Set all widgets in layouts
        GL = QtGui.QGridLayout()
# Grid layout for buttons
        GL.addWidget(self.UpButton, 1, 2)
        GL.addWidget(self.DownButton, 3, 2)
        GL.addWidget(self.LeftButton, 2, 1)
        GL.addWidget(self.RightButton, 2, 3)
        GL.addWidget(self.CWButton, 5, 3)
        GL.addWidget(self.CCWButton, 5, 1)
        GL.addWidget(self.PlotButton, 2, 2)

        GL2 = QtGui.QGridLayout()
# Grid layout for labels
        GL2.addWidget(self.D11, 1, 1)
        GL2.addWidget(self.D12, 1, 2)
        GL2.addWidget(self.D21, 2, 1)
        GL2.addWidget(self.D22, 2, 2)
        GL2.addWidget(self.D31, 3, 1)
        GL2.addWidget(self.D32, 3, 2)
```

```
        VL = QtGui.QVBoxLayout()
# Vertical layout for grid layouts
        VL.addStretch()
        VL.addLayout(GL)
        VL.addWidget(self.AutoPlot)
        VL.addLayout(GL2)
        VL.addStretch()

        HL = QtGui.QHBoxLayout()
# Horizontal layout for canvas and vertical layout
        HL.addWidget(self.canvas)
        HL.addLayout(VL)

        self.setLayout(HL)
        self.showMaximized()
        self.On_Start()
# Start update timer

    def Step(self):
# Function for updating the plot in the UI

        GUI.ax = self.figure.add_subplot(111, projection='3d')
# Create axis on plot
        GUI.ax.hold(True)
        ConfigurePlot()
        StepSize = Speed * Time
        for leg in range(1, 6):
            # Determine desired coordinates of a given leg from x,y vec
            # if (current[2*leg-2] != destination[2*leg-2]):
            DestinationUpdate(leg)
            # current[2*leg-2] = X
            # current[2 * leg - 1] = Y
            reset[leg - 1] = CheckBound(current[2*leg-2], current[2 * l

        for leg in range(1, 6):
            #Reset the legs that need it
            if reset[leg-1] == True:
                ResetLeg(leg)
            X = current[2*leg-2]
            Y = current[2*leg-1]
            # Determine Servo angles for desired position
            Theta, Phi, Alpha = IK(X, Y, Z, leg)
            # Plot result
            Plotleg(Theta, Phi, Alpha, leg, False)
```

```
            # Find position of little orange markers
            x, y = RotateLeg(2.5, 0, leg)
            # Plot little orange markers for home position
            self.ax.plot([x, x], [y, y], [0, -0.05], color='#FFAF00')

        GUI.canvas.draw()
# Refresh canvas
        print "done"

    def UP(self):
# Callback function for Button
        global Y_Vector
        Y_Vector += .1
        if Y_Vector > 0.5:
            Y_Vector = 0.5
        self.D22.setNum(Y_Vector)
        if self.AutoPlot.isChecked():
            self.Step()
        return

    def DOWN(self):
# Callback function for Button
        global Y_Vector
        Y_Vector -= .1
        if Y_Vector < -0.5:
            Y_Vector = -0.5
        self.D22.setNum(Y_Vector)
        if self.AutoPlot.isChecked():
            self.Step()
        return

    def LEFT(self):
# Callback function for Button
        global X_Vector
        X_Vector -= .1
        if X_Vector < -0.5:
            X_Vector = -0.5
        self.D12.setNum(X_Vector)
        if self.AutoPlot.isChecked():
            self.Step()
        return

    def RIGHT(self):
# Callback function for Button
```

```
        global X_Vector
        X_Vector += .1
        if X_Vector > 0.5:
            X_Vector = 0.5
        self.D12.setNum(X_Vector)
        if self.AutoPlot.isChecked():
            self.Step()
        return

    def CW(self):
# Callback function for Button
        global Rot_Vector
        Rot_Vector += 1
        self.D32.setNum(Rot_Vector)
        if self.AutoPlot.isChecked():
            self.Step()
        return

    def CCW(self):
# Callback function for Button
        global Rot_Vector
        Rot_Vector -= 1
        self.D32.setNum(Rot_Vector)
        if self.AutoPlot.isChecked():
            self.Step()
        return

    def On_Timer(self):
        """Executed when the timer runs out"""
        self.Step()
# Automatically take the next step
        print "tick"

    def On_Start(self):
        """This module is called to start the timer and multithreading
        # Start and init timer
        self.timer = QtCore.QTimer()
        self.timer.timeout.connect(self.On_Timer)
        self.timer.start(Time*1000)

        # Start the necessary services
        for leg in range(1, 6):
            X, Y = Vector(-X_Vector, -Y_Vector, leg, -Rot_Vector, True)
            destination[2 * leg - 2] = X
```

```
            destination[2 * leg - 1] = Y
        # End of class GUI


def DestinationUpdate(leg):
    """This module updates the global update array"""
    # X, Y = Vector(-X_Vector, -Y_Vector, leg, -Rot_Vector,True)
    # The stepsize should now be added to the current array in the dire
    current[2 * leg - 2] = destination[2 * leg - 2]
    current[2 * leg - 1] = destination[2 * leg - 1]
    global StepSize
    StepSize = math.sqrt(X_Vector**2+Y_Vector**2+Rot_Vector**2)
# Absolute stepsize from input vectors
    x, y = Vector(-X_Vector, -Y_Vector, leg, -Rot_Vector, False)
    destination[2 * leg - 2] = x + current[2 * leg - 2]
    destination[2 * leg - 1] = y + current[2 * leg - 1]

# Functions from inverse kinematics calculations

def Plotleg(Theta, Phi, Alpha, leg, Dim):
    ""
    "This␣function␣plots␣a␣leg␣after␣the␣inverse␣kinematics␣are␣calcula
    # Call the translation function
    offsetx, offsety = TranslateLeg(leg)

    # Plot segment A
    if Dim == True:
        plotColour = '#FFAFAF'
    else:
        plotColour = '#FF0000'
    LegX = [offsetx, offsetx + A * math.sin(Theta)]
    LegY = [offsety, offsety + A * math.cos(Theta)]
    LegZ = [Z_body, Z_body]
    GUI.ax.plot(LegX, LegY, LegZ, plotColour)

    # Plot segment B
    if Dim == True:
        plotColour = '#AFFFAF'
    else:
        plotColour = '#00FF00'
    LegX = [offsetx + A * math.sin(Theta), offsetx + A * math.sin(Theta
    LegY = [offsety + A * math.cos(Theta), offsety + A * math.cos(Theta
    LegZ = [Z_body, Z_body + B * math.cos(Phi)]
    GUI.ax.plot(LegX, LegY, LegZ, plotColour)
```

```
    # Plot segment C
    if Dim == True:
        plotColour = '#AFAFFF'
    else:
        plotColour = '#0000FF'
    LegX = [offsetx + A * math.sin(Theta) + B * math.sin(Phi) * math.si
            offsetx + A * math.sin(Theta) + B * math.sin(Phi) * math.si
                Phi + Alpha - math.radians(90)) * math.sin(Theta)]
    LegY = [offsety + A * math.cos(Theta) + B * math.sin(Phi) * math.co
            offsety + A * math.cos(Theta) + B * math.sin(Phi) * math.co
                Phi + Alpha - math.radians(90)) * math.cos(Theta)]
    LegZ = [Z_body + B * math.cos(Phi), Z_body + B * math.cos(Phi) + C
    GUI.ax.plot(LegX, LegY, LegZ, plotColour)
    # print (LegX[1], LegY[1], LegZ[1]), "\n\r"
    return

def IK(X, Y, Z, leg):
    ""
    "This␣function␣calculates␣the␣servo␣positions␣required␣for␣specific
    # Call the translatiom function
    Theta = math.atan2(X, Y)  # Calculate Theta
    # print "Theta", leg, "=", math.degrees(Theta)
    ###Calculate Phi & Alpha:
    x1 = A * math.sin(Theta)
    y1 = A * math.cos(Theta)
    z1 = Z_body
    Reach = math.sqrt((X - x1) ** 2 + (Y - y1) ** 2)
# Horizontal Distance from Joint 2 to foot
    TwoToFoot = math.sqrt((X - x1) ** 2 + (Y - y1) ** 2 + (Z - z1) ** 2
# Absolute Distance from Joint 1 to foot
    # print("TwoToFoot")
    # print(TwoToFoot)
    x = (B ** 2 + C ** 2 - TwoToFoot ** 2) / (2 * B * C)
    # print(x)
    d = math.acos(x)
    # print("d")
    # print(math.degrees(d))
    Alpha = 270 - math.degrees(d)
    Alpha = math.radians(Alpha)
    # print "Alpha", leg, "=", math.degrees(Alpha)
    c = math.asin(C * math.sin(d) / TwoToFoot)
    # print("c")
    # print(math.degrees(c))
```

```
    e = math.atan2(Reach, Z_body - Z)
    # print("e")
    # print(math.degrees(e))
    Phi = 180 - math.degrees(c) - math.degrees(e)
    Phi = math.radians(Phi)
    # print "Phi  ", leg, "=", math.degrees(Phi)
    return Theta, Phi, Alpha

def Rotate(xi, yi, angle):
    ""
    "This␣function␣rotates␣the␣coordinates␣of␣a␣specific␣point␣around␣t
    Angle = math.radians(angle)
    xo = xi * math.cos(Angle) - yi * math.sin(Angle)
    yo = xi * math.sin(Angle) + yi * math.cos(Angle)
    return xo, yo

def RotateLeg(x, y, leg):
    ""
    "This␣function␣rotates␣the␣coordinates␣of␣a␣specific␣leg␣to␣its␣req

    Angle = math.radians((leg - 1) * 72)
    X = x * math.cos(Angle) - y * math.sin(Angle)
    Y = x * math.sin(Angle) + y * math.cos(Angle)
    return X, Y

def TranslateLeg(leg):
    ""
    "This␣function␣is␣called␣from␣the␣plotleg␣function␣to␣set␣the␣neces
    "It␣calculates␣the␣offset␣distance␣from␣shoulder␣joint␣to␣robot␣cen
    x = Radius * math.cos(math.radians(72 * (leg - 1)))
    y = Radius * math.sin(math.radians(72 * (leg - 1)))
    return x, y

def Vector(x, y, leg, Rot, Offset):
    ""
    "This␣function␣determines␣the␣required␣position␣of␣the␣foot␣given␣a
    # First do the translation vector
    Angle = math.atan2(y, x)
    Magnitude = math.sqrt(x ** 2 + y ** 2)
# Calculate actual magnitude
    if (Magnitude > 0.5):  # Limit magnitude to 0.5
        Magnitude = 0.5
    if Offset:
        # Find Neutral position of each leg:
```

```
            NeutralX, NeutralY = RotateLeg(2.5, 0, leg)
            # Move leg to the edge of the robot:
            a, b = RotateLeg(Radius, 0, leg)
            NeutralX = NeutralX - a
            NeutralY = NeutralY - b
        else:
            NeutralY = 0
            NeutralX = 0

        X = NeutralX + Magnitude * math.cos(Angle)
        Y = NeutralY + Magnitude * math.sin(Angle)

        # Now the rotation part
        offsetx, offsety = TranslateLeg(leg)
    # Move system origin to robot center
        X = X + offsetx
        Y = Y + offsety
        X, Y = Rotate(X, Y, -10 * Rot)  # Rotate around origin
        X = X - offsetx
        Y = Y - offsety

        if Offset:
            current[2 * leg - 2] = X
            current[2 * leg - 1] = Y

        return X, Y

    def CheckBound(x, y, leg):
        ""
        "This␣function␣checks␣if␣a␣leg␣needs␣to␣be␣reset␣by␣determining␣whe
        NeutralX, NeutralY = RotateLeg(1.5, 0, leg)
        deltaX = NeutralX - x
        deltaY = NeutralY - y
        absDist = math.sqrt(deltaX**2+deltaY**2)
        if absDist > 0.51:
            return True
        else:
            return False

    def ResetLeg(leg):
        ""
        "This␣function␣resets␣a␣leg␣to␣the␣postion␣furthest␣in␣the␣direction
        #Pick up the leg
        Z = 0.5
```

```
    X = destination[2*leg-2]
    Y = destination[2*leg-1]
    # Determine Servo angles for desired position
    Theta, Phi, Alpha = IK(X, Y, Z, leg)
    # Plot result
    Plotleg(Theta, Phi, Alpha, leg, True)
    # plt.pause(0.5)
    # GUI.canvas.draw()
    #Move to reset position
    #Angle =
    #x =
    #y = -(current[2 * leg - 1])
    X, Y = Vector(X_Vector, Y_Vector, leg, Rot_Vector, True)
    current[2*leg-2] = X
    current[2*leg-1] = Y
    destination[2 * leg - 2] = X
    destination[2 * leg - 1] = Y
    Theta, Phi, Alpha = IK(X, Y, Z, leg)
    Plotleg(Theta, Phi, Alpha, leg, True)
    # GUI.canvas.draw
    #Put leg down
    Z = 0
    Theta, Phi, Alpha = IK(X, Y, Z, leg)
    Plotleg(Theta, Phi, Alpha, leg, False)
    return

def ConfigurePlot():
    ""
    "This␣function␣creates␣a␣3D␣plot␣and␣configures␣axes␣and␣labels"
    blank = [0, 0]
    GUI.ax.plot(blank, blank, blank, label="Segment␣A", color='#FF0000'
    GUI.ax.plot(blank, blank, blank, label="Segment␣B", color='#00FF00'
    GUI.ax.plot(blank, blank, blank, label="Segment␣C", color='#0000FF'
    GUI.ax.plot(blank, blank, blank, label="Chassis", color='#AF00FF')
    GUI.ax.plot(blank, blank, blank, label="Feet␣zone", color='#FFAF00'

    p = Circle((0, 0), Radius, color='#AF00FF')
    GUI.ax.add_patch(p)
    art3d.pathpatch_2d_to_3d(p, z=Z_body, zdir="z")

    GUI.ax.legend()
    GUI.ax.set_xlabel('X')
    GUI.ax.set_ylabel('Y')
    GUI.ax.set_zlabel('Z')
```

```
        GUI.ax.axis([-2.5, 2.5, -2.5, 2.5])
        GUI.ax.set_zlim3d(0, 5)

if __name__ == '__main__':
# Start of main application
        app = QtGui.QApplication(sys.argv)
        main = GUI()
# Start a GUI instance
        main.show()

        sys.exit(app.exec_())
# Stop when GUI is closed
```

26. Hardware, software and operating system requirements

The smartphone application requires Android 6 or above to function

27. Software user guide

The user does not interact with the software.

28. Software acceptance test procedure

Although there is a large software component to the project, the output is purely mechanical. All of the results are therefore pass/fail based on the specification.

29. Experimental data

All of the experimental data can be found in part 4 of this report.