HANOI UNIVERSITY OF SCIENCE & TECHNOLOGY

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

--------------------------***--------------------------

PROJECT II
ROBOT VACUUM CLEANER

Instructor: PhD. Nguyen Thanh Huong

Student: Nguyen Thai Nguyen

Student id: 20192241

Hà Nội-20/3/2023

Acknowledge

In the process of completing this project, I have received a lot of support and guidance from teachers, school, friends, and family. Therefore, I want to thank everyone.

I would like to thank the teachers specializing in automation for imparting useful knowledge to me so that I have a basis to carry out my thesis. I would like to sincerely thank Ms. Nguyen Thanh Huong for her guidance and suggestions so that I can complete the project, how to conduct research and edit for me during the research process.

Due to my limited knowledge, in the process of completing this project, I inevitably made mistakes, and I look forward to receiving suggestions from teachers so that I can learn from experience and complete better.

Abstract

A vacuum-cleaning robot is to automatically clean floors by efficiently and systematically removing dirt and debris. It has the ability to move autonomously while avoiding obstacles, demonstrating safety and precision during operation. The robot can be equipped with various devices such as ultrasonic sensors, infrared sensors, MPU6050 angle sensors, protective and charging circuits, stepper motors, and BLDC motors. These features enable the robot to detect obstacles, avoid falling downstairs, and recharge its battery automatically when needed.

The vacuum-cleaning robot is used to clean floors quickly and save time, making it easier for humans to perform their tasks.

Common keywords when discussing vacuum-cleaning robots include: STMf103c8t6 , BLDC Stepper motors, Ultrasonic sensors, Robot vacuum cleaner

Table of Contents

Chapter 1 Introduction

1. Reasons for choosing the topic.

Robots are becoming more and more popular in many different industries and the household cleaning industry is no exception. One of the types of robots that are useful in cleaning the house is a vacuum cleaner. This type of robot is designed to automatically clean floors by effectively and systematically removing dirt and debris. The robot can move around the room by itself, avoiding obstacles and ensuring safety and accuracy during operation.

Robot vacuum cleaners are often equipped with various devices to help them operate efficiently. These may include ultrasonic sensors, infrared sensors, MPU6050 angle sensors, protection and charging circuits, stepper motors and BLDC motors. With these features, the robot can detect obstacles, avoid falling downstairs, and automatically recharge the battery when needed.

The benefits of using a robot vacuum cleaner are many. First and foremost, it saves time and effort for homeowners, who no longer have to spend hours vacuuming the floors themselves. It can also clean more thoroughly and effectively than an operator, as it is designed to cover every inch of the floor and can be easily moved around obstacles. In addition, many robot vacuums can be controlled via a smartphone app or voice assistant, making them easy and convenient to use.

Overall, the robot vacuum cleaner is a useful and innovative tool for household cleaning. Its automatic operation and advanced features make it an attractive choice for those who want to make their cleaning job more efficient and easier.

2. Purpose

- Robot can move automatically with zigzag trajectory.

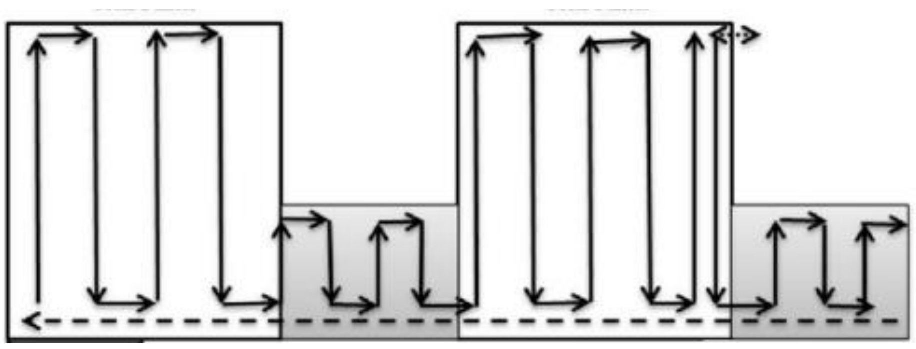


Figure 1: zigzag trajectory line

- can sense the low battery capacity and return to the charging point.
- Can avoid obstacles and avoid tripping stairs.
- can be controlled via smartphone by Bluetooth.
- Can suck up stubborn dirt with strong suction.
- Robot size must be suitable for household use.
- Uptime long enough to clean for hours at a time.

In this project, I started to do basic research and simulations about robots, thereby creating a foundation for more extensive research on this topic, and the aim of the project is to be as complete as possible with a topic that but the problem is still new and full of challenges.

3. Research Methods

- The robot model is designed based on the round vacuum cleaner model of the best brands today.

- Using microprocessor stm32f103c8t6 as the central processor to control modules and sensors.

- HC-05 for data transmission over the phone

- Module A4988 is used to receive commands from STM32 to control the motor

- Ultrasonic sensor has the function to help the robot avoid obstacles and prevent the robot from falling and damaging the robot.

- the robot uses a BMS to manage and charge the battery safely and efficiently.

- MPU6050 sensor is used so that the robot can rotate perpendicularly.

Chapter 2 Overview of robot vacuum cleaner

1. Robot development history

The development history of robot vacuum cleaners. Before there was no robot vacuum cleaner, people had to use tools to clean the house, this was a common method. But with the continuous development of society, the needs of people in daily life are increasing. Cleaning the house takes a lot of effort if you must clean in large areas.

By 1901, motorized vacuum cleaners were created. This is a device that uses a motor to pump air to create a vacuum that picks up dust. But after a period of use, it was found that the vacuum cleaner caused many problems such as bulky structure, direct plugging in, so the operator had difficulty because of limited mobility. That's why robot vacuum cleaners were invented, in the early years of robot vacuum cleaner development, it was not known much, and the price was high at that time, so it was not popular in the market. Electrolux a Swedish company has launched the first robot vacuum cleaner model to appear on an online program called Electrolux Trilobite. Introduced in 2001, the Trilobite contains a removable brush and vacuum cleaner with ultrasonic sensor obstacle avoidance. There are many different types of robot vacuums on the market today, such as

Roomba robot vacuum cleaner:

Figure 2 Roomba robot

Chapter 3 Specification and Theory

1. Architecture design

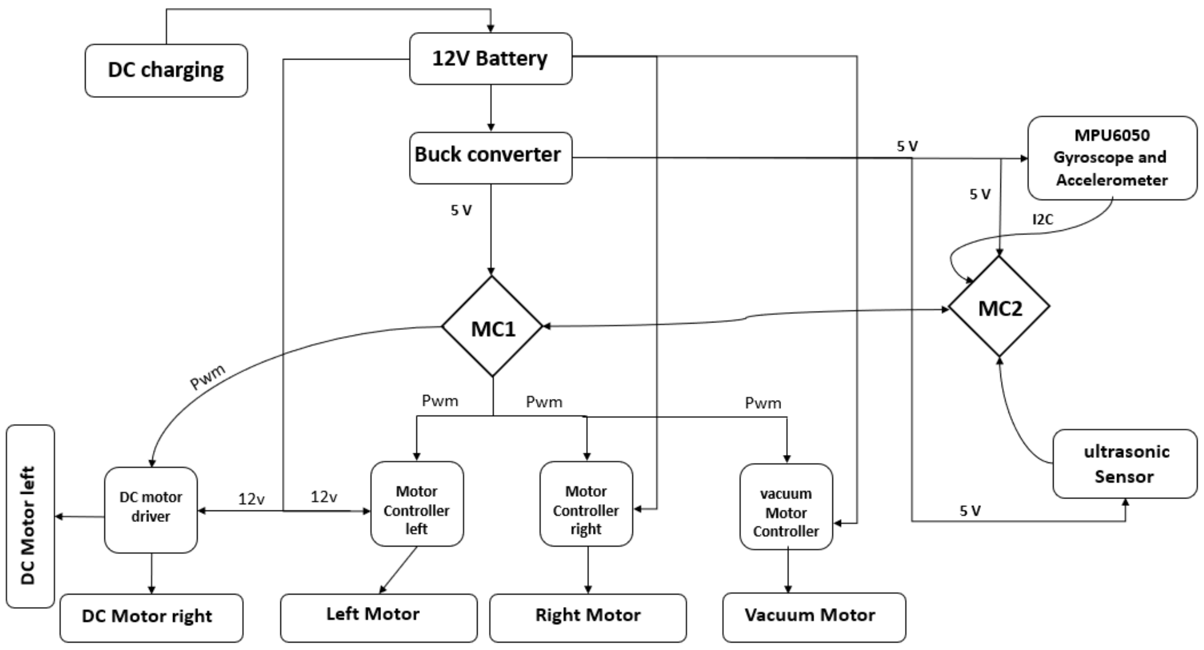


Figure 3 block diagram.

- The battery charging circuit is fixed on the robot, when the robot is almost exhausted, it can be charged immediately.
- Power supply to motors with 12Voltage, and the power supply to the first and the second Microcontroller, sensors are 5Voltage.
- Describe the process: the functions of ultrasonic sensors are used to determine the obstacles and it send the signal to Second Microcontroller and the enable signal is sent to the First Microcontroller, First Microcontroller will send the signal about direction and speed to Motor Controller left and Motor controller right to turn left or turn right. Because the robot run follow the zigzag trajectory line, it needs to turn 90 degrees perpendicular. Therefore, the MPU6050 will measure and calculate the angle.

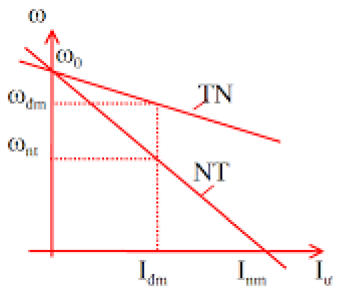
- The Vacuum Motor is BLDC motor, it is controlled by Vacuum Motor Controller to cleaning the dirt.

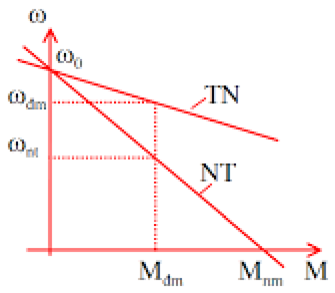
2. BLDC Motor

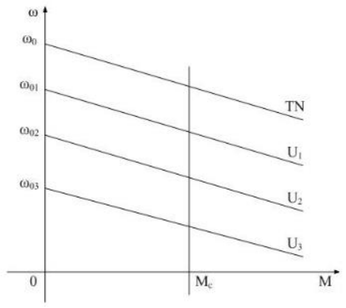
BLDC (brushless DC) motors have several advantages over other types of motors: High Efficiency, High Power Density, Low Maintenance, Smooth Operation, Precise Speed Control

The motor factor k is a way to consider the increased torque necessary due to start up, impacts and irregularities from the driven machine. For electrical motors which drive fans the value of k is 1.4

❖ mechanical property equation



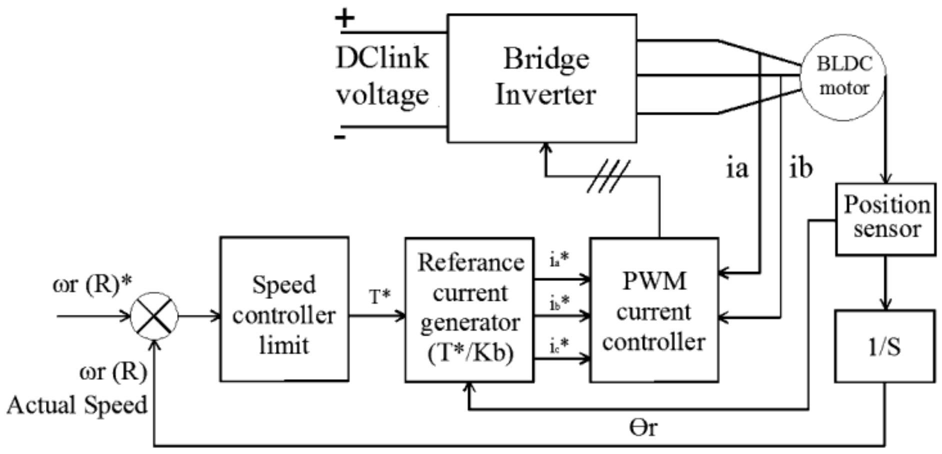




Speed: $\omega = \frac{U_{ư}}{K\phi} - \frac{R_f + R_u}{K\phi} I$

Torque: M = K.I.ϕ

❖ BLDC control speed



1. Ultrasonic senser

The human ear can detect frequencies between 20 and 20 000 Hz . Anything above 20 000 Hz is called infrasound and anything below 20 Hz is called infrasound.

The ultrasonic sensor uses a frequency of approximately 40 000 Hz. The sensor has a transmitter that emits sound waves and a receiver that detects echoes from an object. Transmitter and receiver together are a type of transducer. A converter that converts physical energy into electricity and

opposite. The transmitter is like a speaker, it has a membrane that produces sound waves. Electrical energy causes the membrane to move, creating waves. The receiver works in a similar way, but in reverse. The energy from the sound waves causes the membrane to move creating an electrical signal. Sound waves have a certain scattering angle, which depends on the size of the film. Figure 14 shows a transmitter, “T” on the sensor and a receiver, “R” on the sensor.

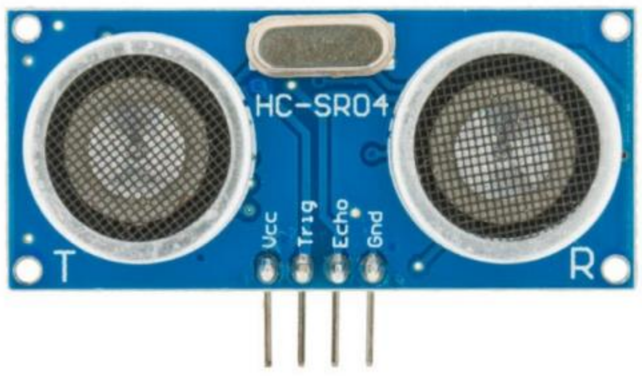


Figure 4: ultrasonic senser

With help of the software, the distance can be calculated. This is done by taking the time from when the transmitter transmits to when the receiver gets a sound wave back. With a temperature of 20 degrees, the velocity of sound is 340 m/s. The distance can be calculated by: $L = \frac{v.t}{2}$

Where t is the time and v is the velocity. The formula is divided by 2 because the soundwave goes to the obstacle and back. Ultrasonic sensors have different applications. A transmitter can only make one type of sound wave, which is going to have a certain scattering angle and certain range. This angle and range, limits an ultrasonic sensor to a specific application.

2. Fan

Centrifugal force on a particle is described with the equations below

F = $\frac{mv^2}{r} = m.r.w^2$

F = m.a

Combined

$$a = \frac{v^2}{r} = rw^2$$

where:

F is the force directed towards the rotation center [N]

M is the mass of the particle [kg]

v is the tangential velocity [m/s]

w is the angular velocity [rad/s]

r is radius [m]

a is radial acceleration [m/s^2]

the average

the average acceleration between the inner radius Ri and the outer radius Ry over the distance Ry-Ri results in a change in velocity which results in pressure (p[pa]) change. According to Bernoulli's equation for certain conditions. v is the radial velocity m/s.

p1 – p2 = $\frac{v_2^2}{2} - \frac{v_1^2}{2} = \frac{1}{2}(v_2^2 - v_1^2)$

v1 is the velocity of inner radius.

v2 is the velocity of outer radius.

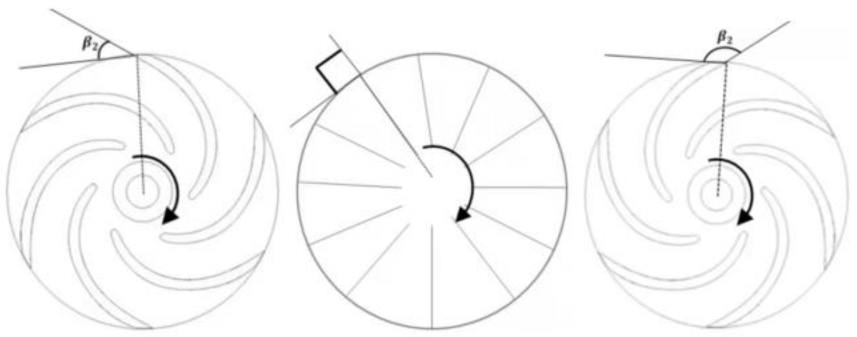


Figure 5 fan

3. **Second controller**

ATmega328P is an 8-bit microcontroller produced by Atmel, widely used in embedded electronics applications and Arduino projects. Here are some advantages of ATmega328P, Fast processing speed, Large memory, Diverse features, Energy efficiency, Easy to use, Reasonable cost

4. **Stepper motor**

Step motors have several advantages over other types of motors: Precise Control, High Torque at Low Speeds, Simple Control Circuitry, No Feedback Required, Low Cost

5. **Battery**

The advantages of Lithium-ion 18650 battery include: High capacity, Long lifespan, no self-discharge, Safety overvoltage, minimizing the risk of electric shock and explosion. Easy to use and replace, environmentally friendly.

6. **First controller:**

The STM32F103C8T6 microcontroller is an ARM Cortex-M3 based microcontroller, developed by STMicroelectronics. It is one of the most popular microcontrollers used in embedded applications, because it has several advantages, such as, High Performance, Large Memory, Energy Efficient Multitasking, Flexible Communication: The STM32F103C8T6 microcontroller has many communication ports such as USART, SPI, I2C, ADC, DAC, PWM... allowing it to connect to various devices.

Chapter 4 Control algorithm

1. The problem of the moving trajectory of the robot

Have many moving trajectories for mobile robot vacuum cleaners such as zigzag, spiral, etc. to vacuum the entire floor. In which, the zigzag-shaped trajectory is suitable for the empty floor, with few obstacles. In this article, we only deal with the robot’s zigzag trajectory.

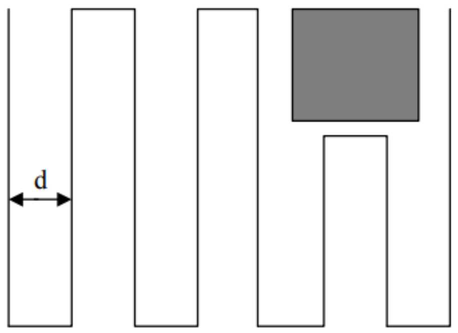


Figure 6 the zigzag trajectory

2. Modeling

The robot vacuum cleaner uses a two-wheeled mobile robot model. Hypothesis two wheels rolling without slipping. The forward kinematics of the robot vacuum cleaner and the origin with reference trajectories are as shown below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\emptyset} \end{bmatrix} = \begin{bmatrix} cos\emptyset & 0 \\ sin\emptyset & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

Where C(x,y) is the center of the robot in Cartesian coordinates system, $\emptyset$ is the rotation angle of the robot. v,w are the forward and angular velocity of the center of vacuum cleaner robot.

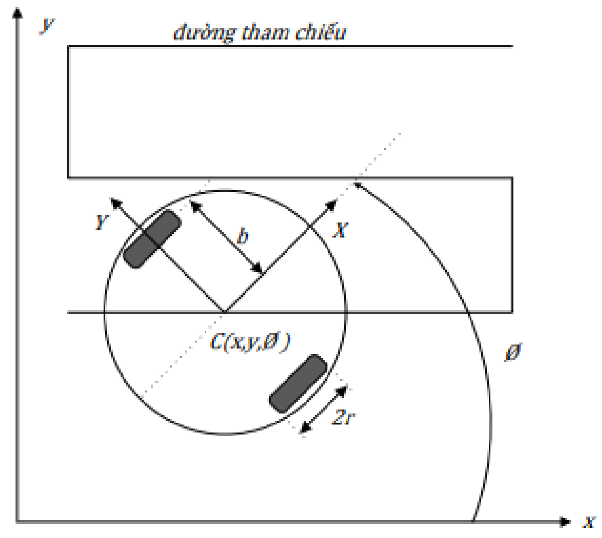


Figure 7 Modeling of vacuum cleaning robot

The relationship between v, w and the angular velocity of two wheels:

$$\begin{bmatrix} w_{rw} \\ w_{lw} \end{bmatrix} = \begin{bmatrix} 1/r & b/r \\ 1/r & -b/r \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

In which w_{rw},w_{lw} are the angular velocity of left and right wheel, b is the distance between two-wheel, r is the wheel radius.

The center of the area to be vacuumed coincides with the center C of the robot. the reference point R move with velocity constantly v_r. Therefore, the control purpose is controlling the center C of robot follow the reference point R. the point R lie on the reference line being the straight line so.

$$\begin{cases} \dot{x_r} = v_r \\ \dot{y_r} = c \\ \dot{\emptyset_r} = 0 \end{cases}$$

And c = constant

The error when follow the orbit.

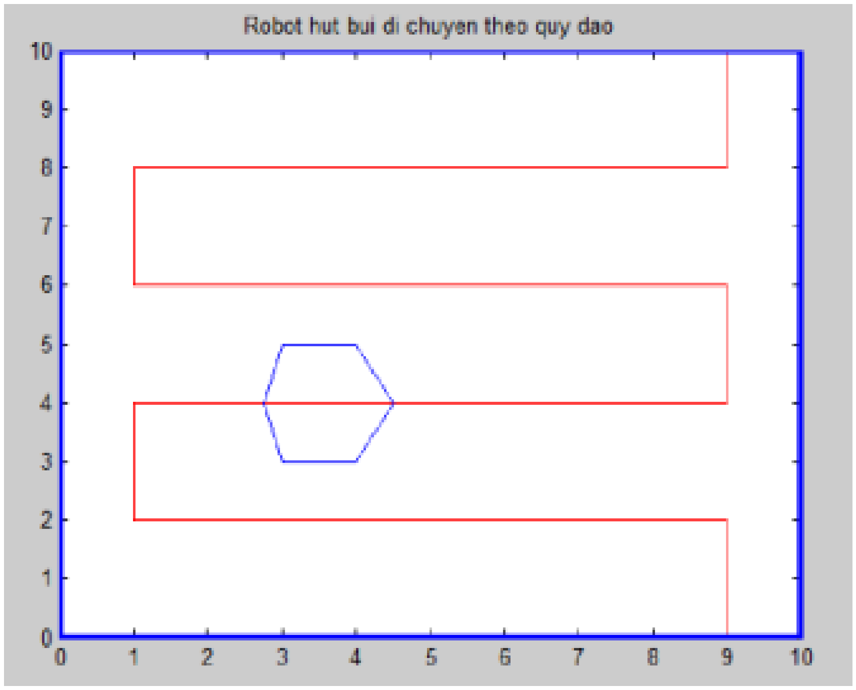
$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} x - x_r \\ y - y_r \\ \emptyset - \emptyset_r \end{bmatrix}$$

e_1, e_2, e_3 are errors according to x, y and the swivel angle $\emptyset$ of center C and the reference point R on the orbit.

The control purpose: $\boldsymbol{e_i} \to 0$,

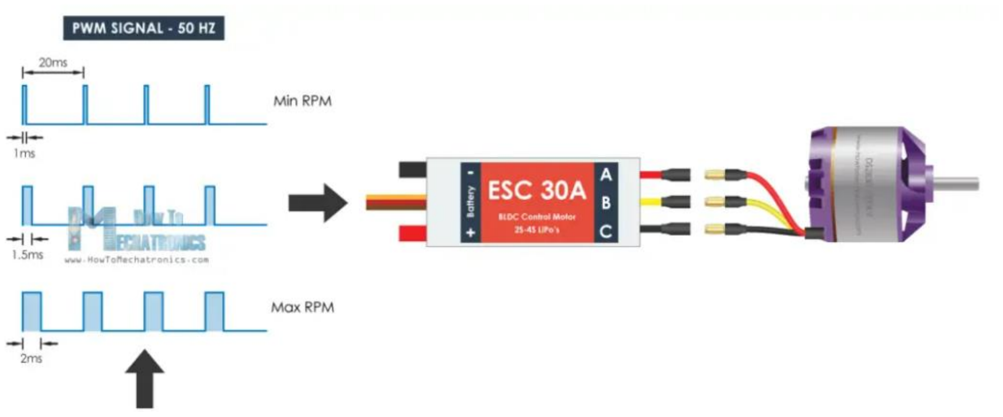
$$\begin{bmatrix} \dot{e_1} \\ \dot{e_2} \\ \dot{e_3} \end{bmatrix} = \begin{bmatrix} v.\cos\emptyset - v_r \\ v.\cos\emptyset - c \\ w \end{bmatrix}$$

However, stepper motor operates at almost exact speed, therefore the errors are almost zero.



Chapter 5 Hardware

1. BLDC motor



- ❖ input pins of ESC30A
  - 2 power cords for the motor
  - power pins for esc (5V and ground)
  - 1 PWM pulse pin for speed control
- ❖ Output pins of ESC: phaseA, phaseB, phaseC.

battery have 3 cell and the output voltage is 11.1V therefore the maximum speed of motor is 11.1×1000KV = 11100RPM.ESCs use the same type of control signal as servo and that's the standard 50Hz PWM signal. we just have to generate the 50Hz PWM signal and depending on pulses width or the high state duration which should vary from 1 millisecond to 2 milliseconds, the ESC will drive the motor from minimum to maximum RPM.

$$F_{PWM} = \frac{F_{timerClock}}{(Prescaler + 1)(counterPeriod + 1)}$$

$$F_{timer\ clock} = 8\,000\,000\ Hz$$

$$F_{PWM} = 50Hz$$

$\rightarrow$ (Prescaler + 1) $\times$ (CounterPeriod + 1) $= \frac{F_{timerClock}}{F_{pwm}} = \frac{8\,000\,000}{50} = 160\,000$

Choose Prescaler = 80 – 1 => CounterPeriod $= \frac{160\,000}{80} - 1 = 2000 - 1$

Choose Duty Cycle $= 6.5\% => $ Pulse Width $= \frac{6.5}{100} \times 2000 = 130$

2. Stepper motor

❖ Speed

The desired maximum velocity of the system can be converted into a wheel rotational speed by using the following equations. System need to move at a speed of 1m/s and the radius of the wheel is 0.0325m. the maximum velocity of the robot is translated into rotational speed of the wheel by using the wheel radius

$$n_{wheel} = \frac{maximum\ velocity}{wheel\ circumference} = \frac{1m/s}{2\pi.0.0325} \times \frac{60\ sec}{1\ min} = 294\ RPM$$

❖ Torque

$$F_{torque} = ma + F_{dynamic} + F_{air}$$

Where is the force from the stepper motor:

$$F_{torque} = \frac{M_{tot}}{r}$$

$F_{dynamic}$ Is the wheel friction.

$$F_{dynamic} = NC_r = mgC_r$$

Where $C_r = 0.015$

F_{air}Is the air resistance and because of the low speed the air resistance is zero

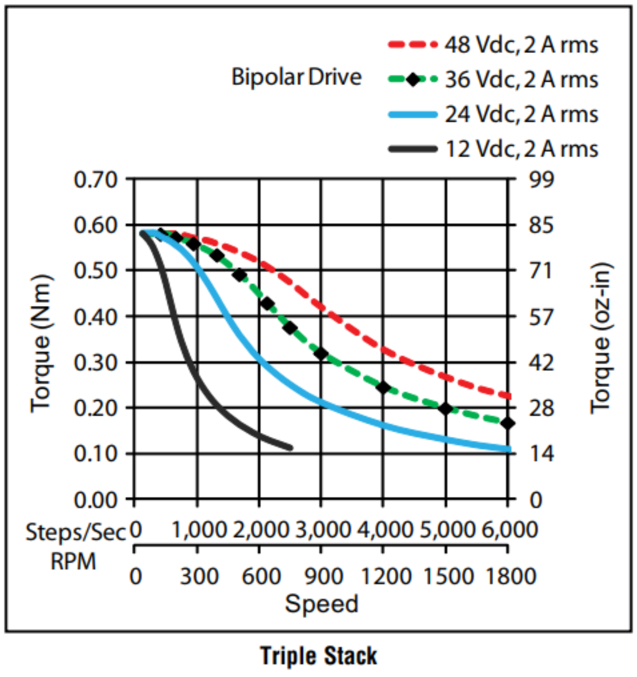


Figure 8 Torque and speed of stepper motor

$M_{tot} \approx 0.25Nm.$ (because speed = 294 RPM)

From the equation 3.11, 3.12 and 3.13 the mass can now be calculated:

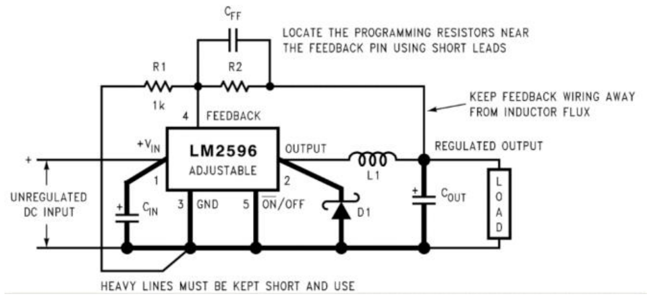
m = $\frac{M_{tot}}{r(a+gC_r)}$ where a = 1 m/s^2, g = 10 m/s^2, r = 0.0325 (m)

$m_{max} = 6.7\ kg$

with the values that has been calculated the maximum mass that one stepper motor can transport is : m = 6.7 kg.

3. DC-DC converter

To provide stable voltage for Microcontroller, sensors, we need to step down voltage from 12-volt input to 5V. to do this, we apply LM2596S-ADJ which are high efficiency adjustable voltage regulators. This is a good power supply, with small size and heat capacity.


C_FF
LOCATE THE PROGRAMMING RESISTORS NEAR THE FEEDBACK PIN USING SHORT LEADS
R1
1k
R2
4 FEEDBACK
KEEP FEEDBACK WIRING AWAY FROM INDUCTOR FLUX
+V_IN
LM2596
ADJUSTABLE
OUTPUT
REGULATED OUTPUT
L1
UNREGULATED DC INPUT
C_IN
3 GND
5 ON/OFF
D1
C_OUT
LOAD
HEAVY LINES MUST BE KEPT SHORT AND USE

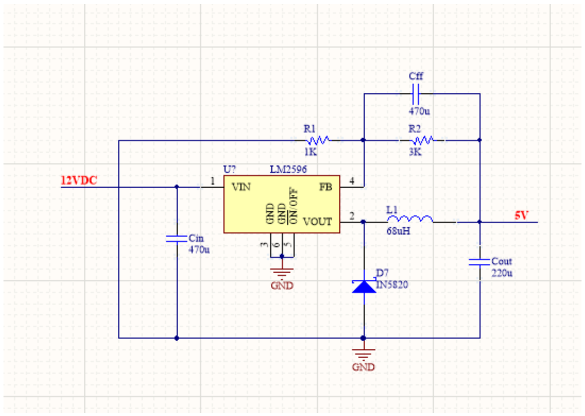


Figure 9 LM2596

at this topology, we must use R1 = 1k Ohm for best quality performance and match the ideal input current of the IC LM2596.

Cin is chosen to be 470uF-50V

Cout = 220uF for transient response. The diode is required to have fast switching characteristics. So we chose Schottky Rectifier 1N5825. The coil for filter application = 68uH.

The output voltage is calculated by using formula: Vout = $V_{ref}\left(1+\frac{R2}{R1}\right)$; and Vref = 1.23V so in order to create 5V at the output then R2 should be 3k Ohm.

C_{FF} is feedforward capacitor which is used to add lead compensation to the feedback loop and increases phase margin for better loop stability. We choose the value = 470uF.

4. Motor driver

Every stepper motor needs a driver card. The driver card that are used for our stepper motors of model A4988. In the figure is shown how to connect the driver card to make it work correctly.

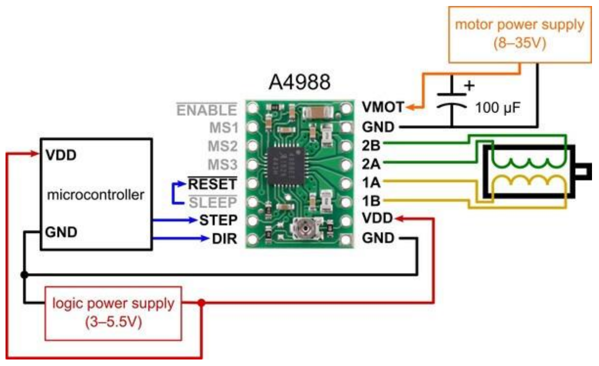


Figure 10 a4988 pins

❖ Adjusting the current

The A4988 allows you to set a target current anywhere between some mA up to a bit less than 2A, this is accomplished by adjusting what is called the Vref (Reference Voltage) when turning the pot on a clockwise direction the Vref voltage will increase and decrease when rotating it counterclockwise.

The actual value of Vref can be calculated using the formula: $V_{ref} = I_{max} \times R_s \times 8$

The maximum current of Motor is 1.2A and we want to run at 80% of its rating:

Imax = 1.2 × 0.8 = 0.96 A

The sensing resistor Rs value of 0.1Ω

⇨ Vref = 0.96 × 8 × 0.1 = 0.768 V.

5. Battery

Continuous discharge current of BLDC motor is: 30A.

The maximum current of two Stepper motor is: 1.2A x 2 = 2.4A

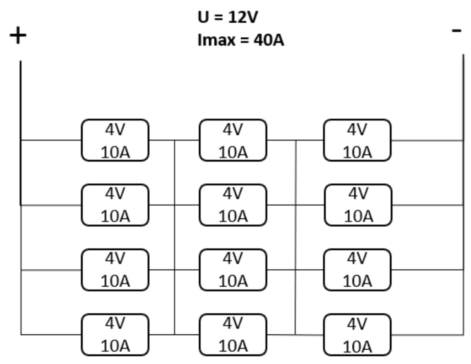
⇨ Continuous discharge current of Battery: Imax > (30+2.4) = 32.4 A
⇨ Choose Imax = 40 A

Moreover, the working voltage of system is: 12V.

⇨ Have two reasons of this requirement:

Name	Advantage	Disadvantage
Lipo-battery	High continuous discharge current, compact size	High price, low capacity, less common in projects than lithium
Lithium 18650	high battery capacity, popular applications in projects,	low continuous discharge current, space occupation

⇨ Choose Lithium 18650 battery:



The maximum current: Imax = 10 x 4 = 40A.
The voltage: U = 4 x 3 = 12V.
battery system capacity: 2800mAh x 4 = 11200mAh.
⇨ P = 11.2 x 12 = 134.4Wh

❖ **The least time the robot can work:**

BLDC motor maximum power consumption:
$P_{bldc} = I_{ESC} \times Vcc = 30 \times 12 = 360$ w
Two Stepper motor maximum power consumption:
$P_{step} = I_{max} \times Vcc \times 2 = 1.2 \times 12 \times 2 = 28.8$ w
motor maximum power consumption:
$P_{DC} = I_{max} \times Vcc \times 2 = 0.12 \times 12 \times 2 = 2.88$ w
The maximum power of system: $P_{sys} = 360 + 28.8 + 2.88 = 391.68\ W$
⇨ Minimum working time = $\frac{134.4 \times 60}{391.68} = 20\ minutes$

6. Battery charging circuit

To charge the battery for the robot vacuum cleaner, it is necessary to have a charging and discharging circuit and protect the battery 3S. To ensure the robot's long-term operation, a large battery capacity and circuitry to protect the battery from overload or extreme contact can lead to fire and explosion. The 3S battery charge and discharge protection circuit is suitable for lithium batteries with a nominal voltage of 3.7 V and a full voltage of 4.2 V.

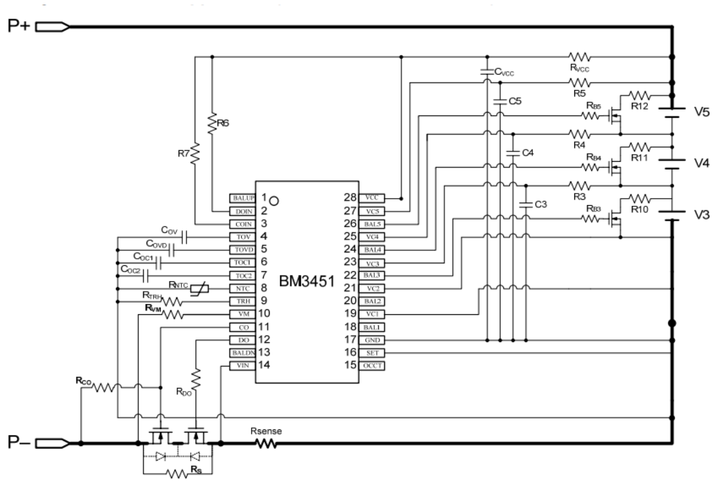


Figure 6（c）**3-cell application (SET be connected to GND)---with balance function**

Figure 11 battery charging structure

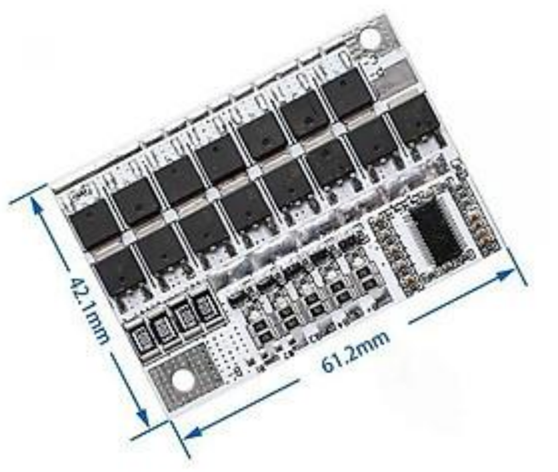


Figure 12 battery charging board.

balanced voltage 12.6V-13.6V

Continuous discharge current: 60A (in the ideal case)

Continuous charging current: 60A

7. Ultrasonic sensers

The robot vacuum cleaner has a total of five ultrasonic sensors. 3 acoustic sensors control if there are any obstacles or walls in the way. 2 ultrasonic sensors are used to replace 2 infrared sensors for simpler construction and to control if there are stairs. Sensor of the same model, HC– Sr04.

Model HC – Sr04 has a measuring range from 2 cm to 400 cm and a total measuring angle of 30 degrees. A measurement test was performed.

Chapter 6 Software

1. Finite state diagram

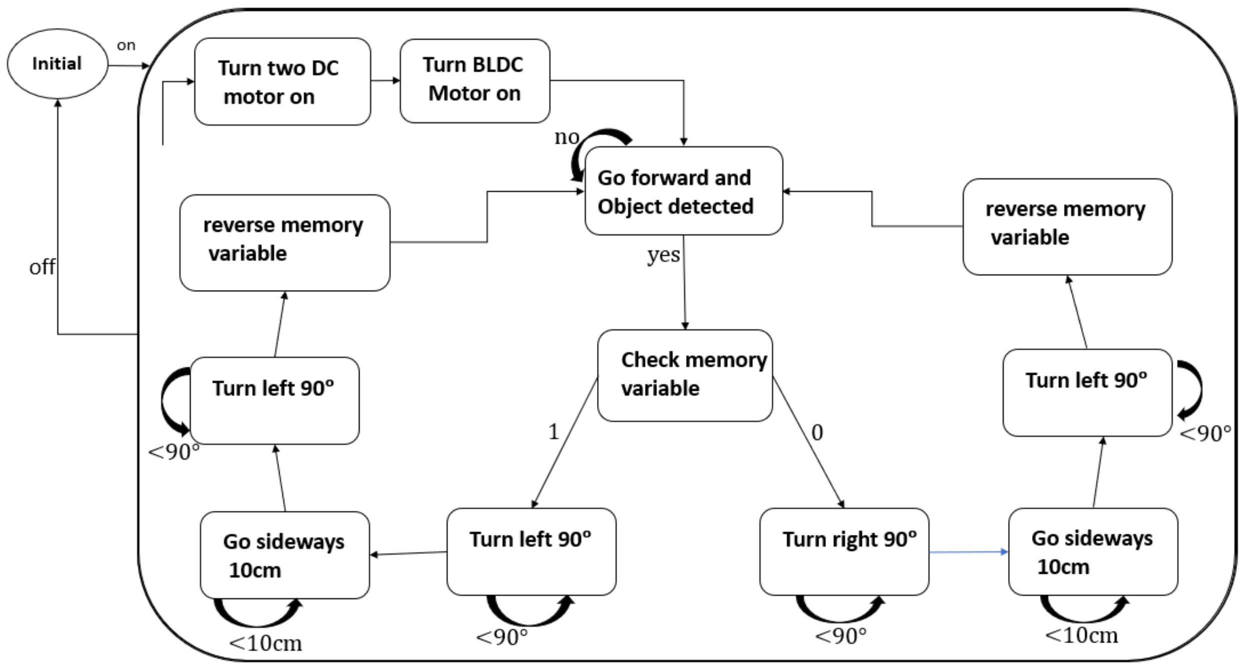


Figure 13 finite state diagram

When robot receive the start signal from user.

First step: is turning two brush Motor and cleaning Motor on (turn two DC Motor and BLDC Motor on).

Second step: is turning two movement motor on and go forward. Check object or stair to save robot. if the result is Low, robot will continue go forward. If the result is High, robot will Check memory variable to turn left or right.

Third step: if the value of memory variable is High, robot will turn left or right 90 degrees, this loop will return until the turn angle of robot equal or higher 90 degree.

Fourth step: after turn 90-degree, robot will go sideways until reach 10cm (this value depends on programmer). To come back the initial direction, robot have to turn 90-degree gain to turn back. At the end, robot continue go forward, however, memory variable needs to reverse to ensure that robot follow the zigzag trajectory line.

2. Design the turning process.

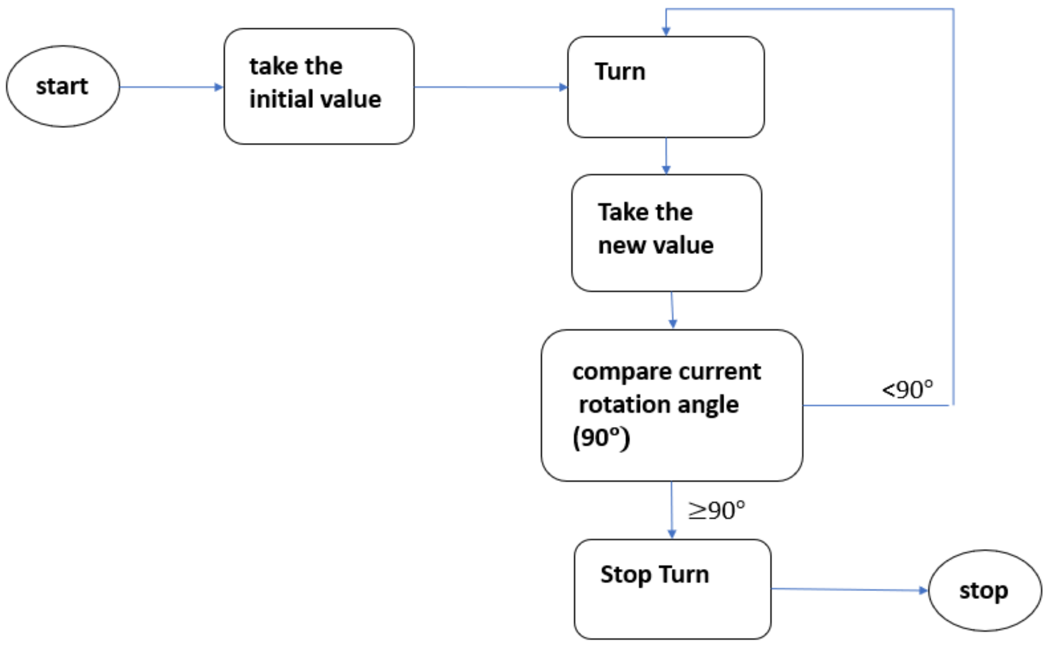


Figure 14 turning process diagram.

After receiving the rotation signal, the program will save the first value as the z-axis position corresponding to the 0-degree angle, then the robot starts to rotate and at the same time takes the new value when the measured value is greater than the value. initial interval corresponding to 90 degrees, the program will stop turning and exit the turning program to continue executing other programs.

Chapter 7 Results and Further advancement

Results

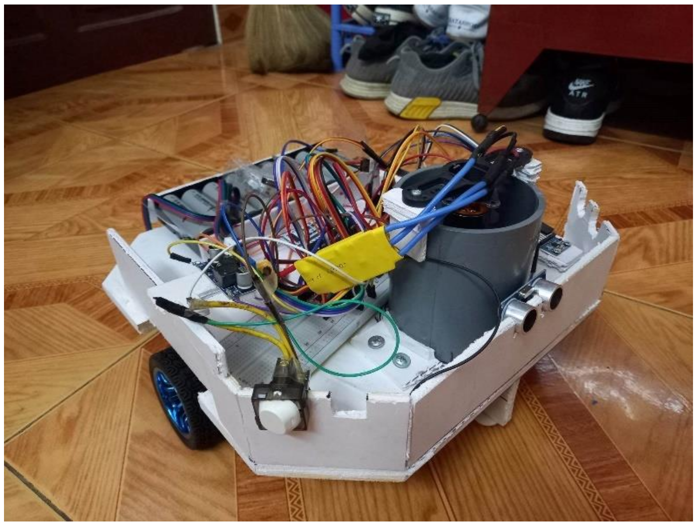


Figure 15 result

Compared with the original goal, the robot has not yet found a way to return to the charging station and connect with the user via smart phone, and its movement is still not fully optimized, and at the same time, some other types of migrations have not been implemented yet. However, it is worth noting that the robot has met the most basic and important requirements.

The current robot can move steadily and identify obstacles to avoid them and can effectively move in a zigzag trajectory line. Besides, the robot has been equipped with the ability to vacuum and can automatically charge, balance, and protect the battery.

Although robots still have many limitations in performing more advanced functions such as connecting with users via smart phones and performing many other types of movement, the progress of robots in meeting the basic requirements of the robot is still limited. original and important was a significant step forward. The development of robots will continue to be promoted to achieve higher advancements in the future.

Further advancements

The results in this project are still limited in many functions. Some features that can be improved in the future to make the robot even better are as follows:

- the robot can return to the charging station

- User can select the functions and control the robot through smart phone.

- the robot can identify water and other obstructions such as electrical wires.

- The robot can perform an additional function of cleaning the house with water.

Some hardware improvements:

- the circuit boards on the test board are not effective when taking up a lot of space and are not safe, it is necessary to do on printed circuits the components must be fixed.

- the motor needs to be re-selected to reduce noise when traveling. Stepper motor with slow travel speed should be replaced by a DC motor with speed feedback by encoder.

REFERENCES

[1] https://www.ti.com/lit/ds/symlink/lm2596.pdf

[2] https://arxiv.org/ftp/arxiv/papers/1412/1412.0591.pdf

[3]https://www.ucg.ac.me/skladiste/blog_13268/objava_56689/fajlovi/Introduction%20to%20Autonomous%20Mobile%20Robots%20book.pdf

[4] http://www.si-semic.com/upload/1460800283.pdf

[5] https://kienthuctudonghoa.com/motor-buoc-la-gi/

[6]https://www.pololu.com/file/0J450/a4988_DMOS_microstepping_driver_with_translator.pdf

[8]https://www.tme.eu/Document/25459777e672c305e474897eef284f74/POLOLU-2128.pdf

[9] https://howtomechatronics.com/tutorials/arduino/arduino-brushless-motor-control-tutorial-esc-bldc/

[10] https://www.youtube.com/watch?v=1RPa16FxR8o

[11] https://ardufocus.com/howto/a4988-motor-current-tuning/

[12] https://eng-resources.charlotte.edu/unccengkit/mechanical/motors/selecting-stepper-motors/

[13] https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf

[14] https://www.youtube.com/watch?v=IiE8skW8btE

APPENDIX

GPIOs OF STM32F103C8T6

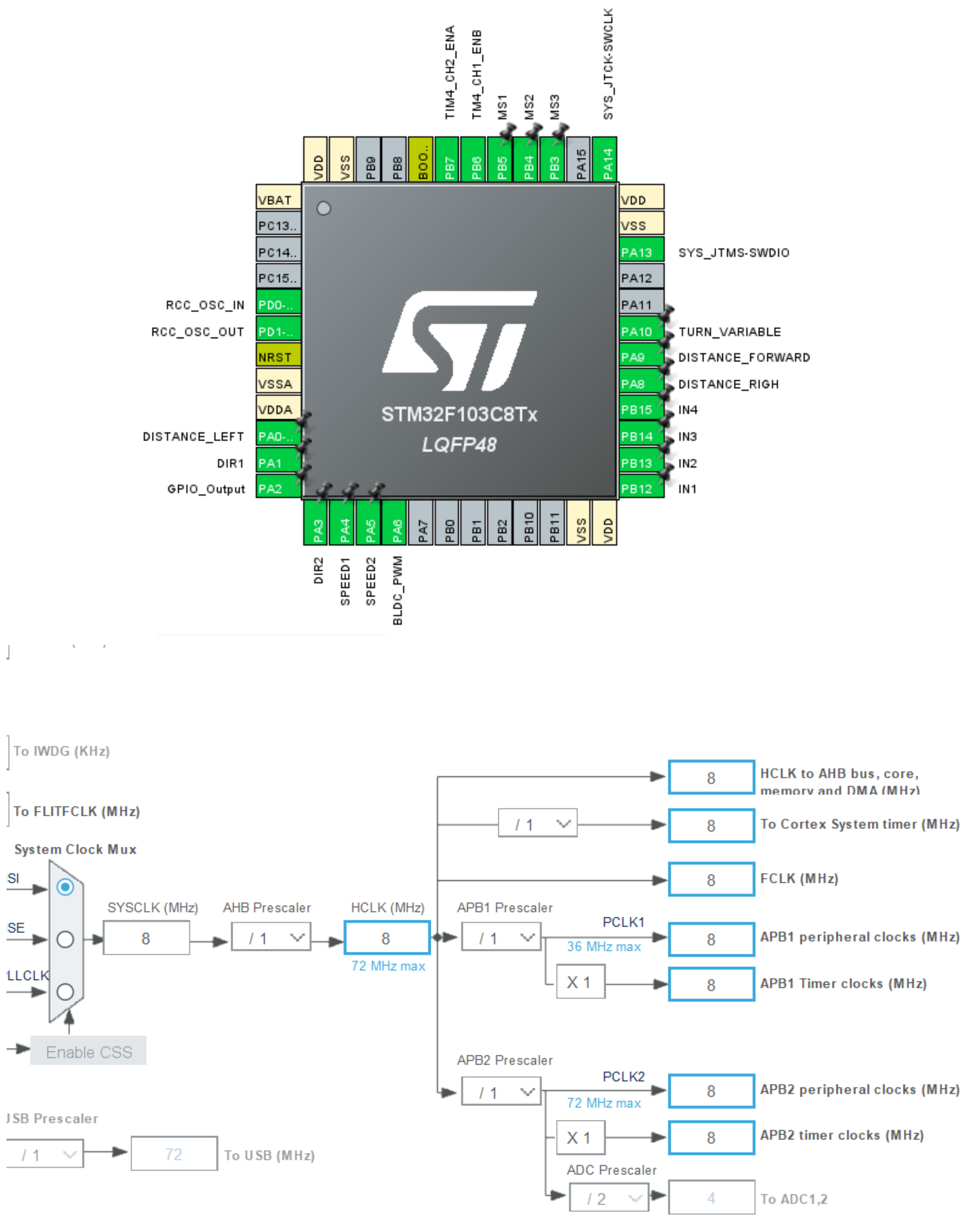
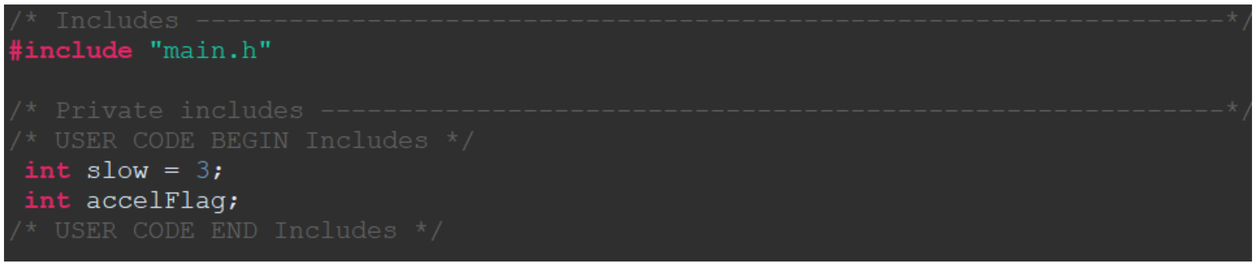
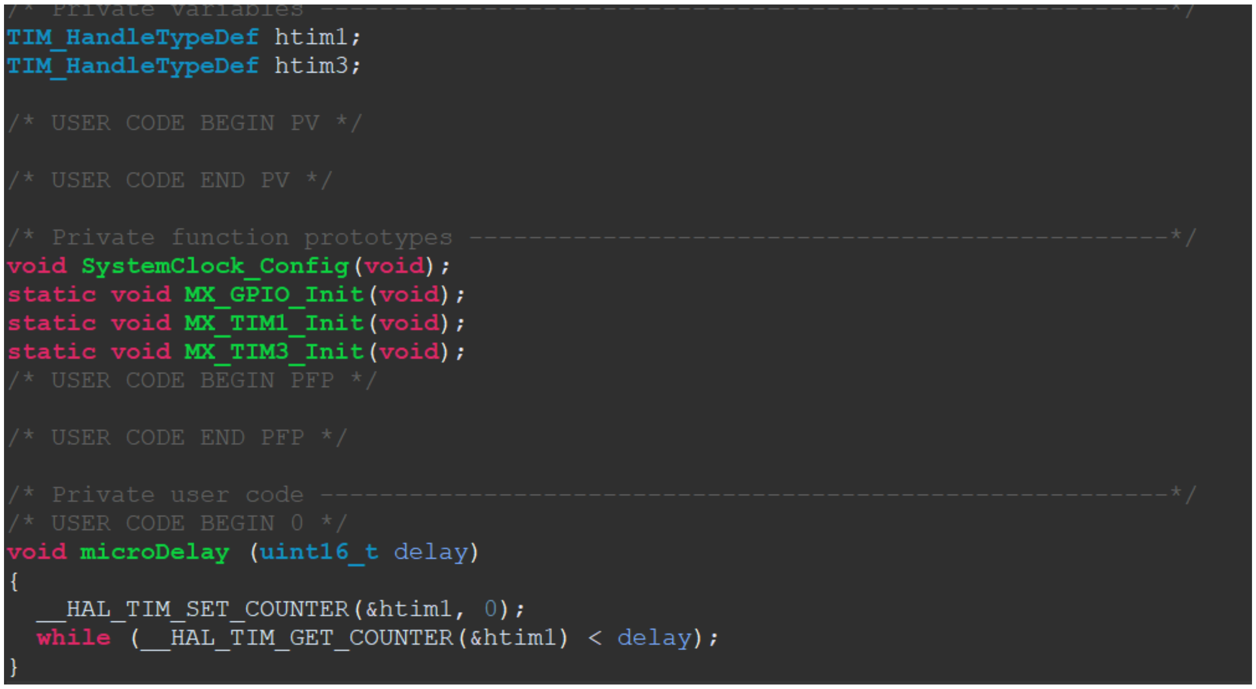


Figure 16 configurations GPIOs of STM32f103c8t6

CODE IN STM32



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

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
 int slow = 3;
 int accelFlag;
/* USER CODE END Includes */
```



```
/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim3;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM3_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void microDelay (uint16_t delay)
{
  __HAL_TIM_SET_COUNTER(&htim1, 0);
  while (__HAL_TIM_GET_COUNTER(&htim1) < delay);
}
```

The program have five functions include:

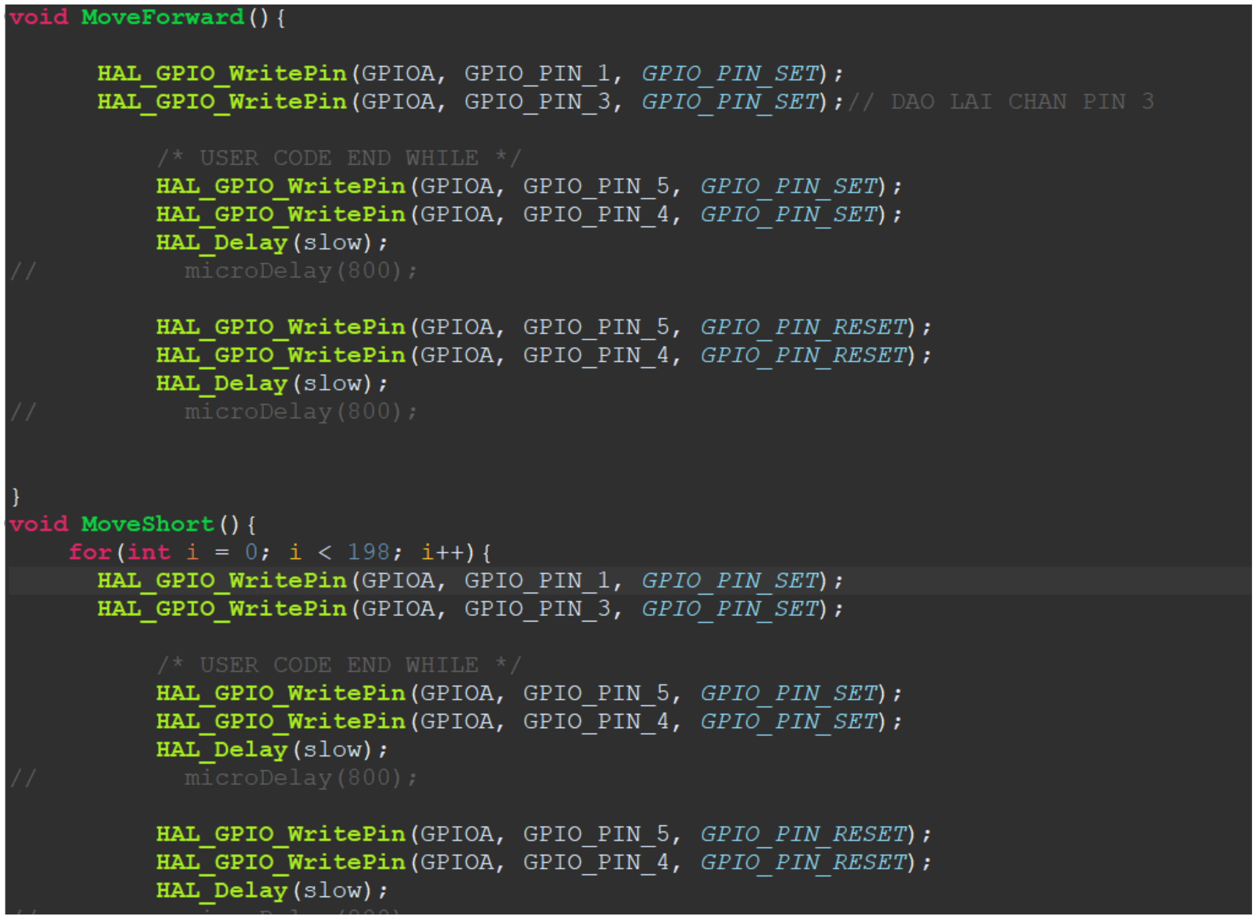
MoveForward(): robot goes straight forward.

MoveShort(): the robot goes straight about 5cm after turning 90 degrees.

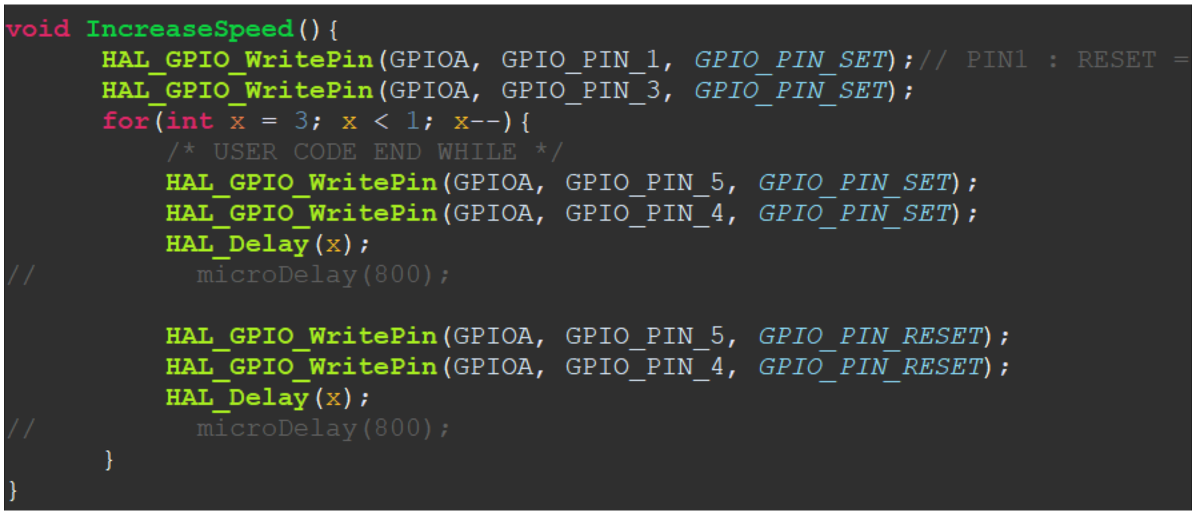
IncreaseSpeed(): the function help robot start smoothly.

TurnLeftEye(): turn left 90 degree.

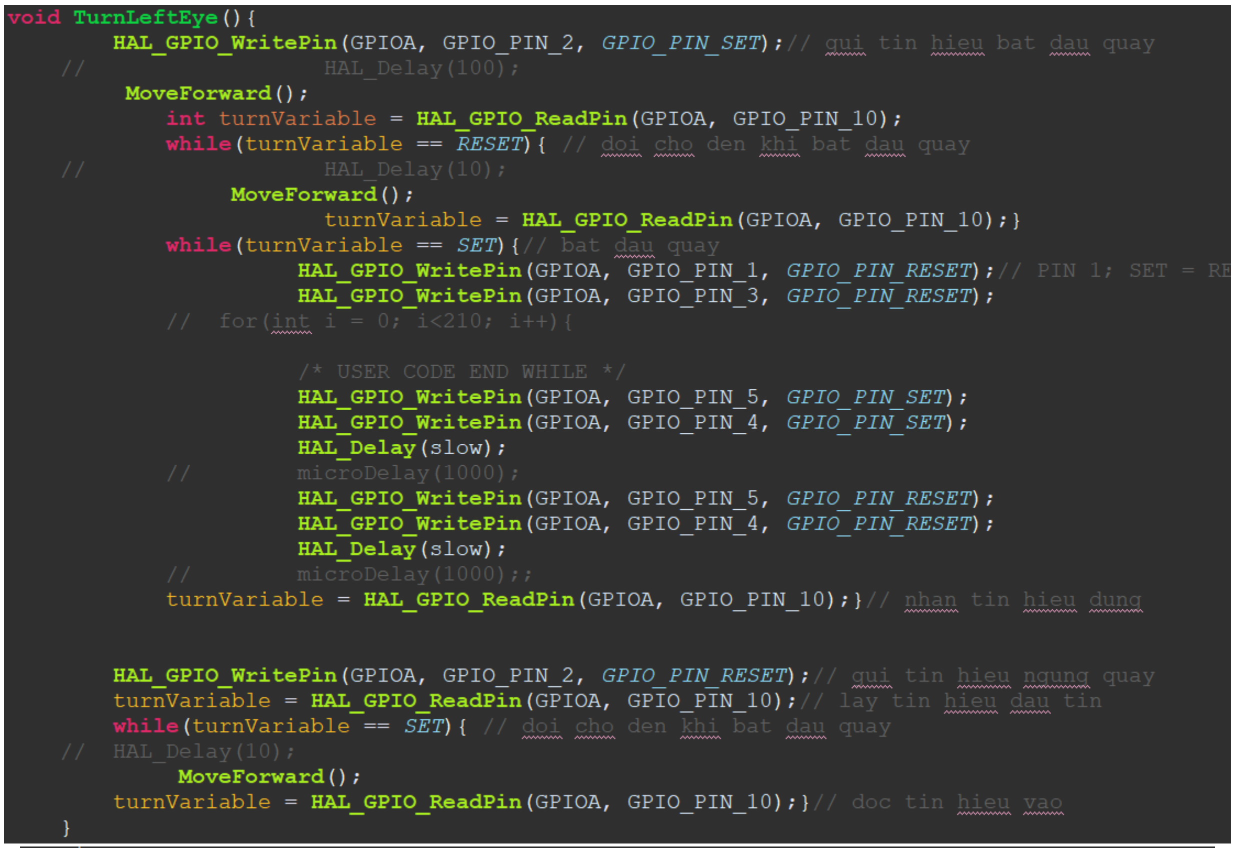
TurnRightEye(): turn right 90 degree.



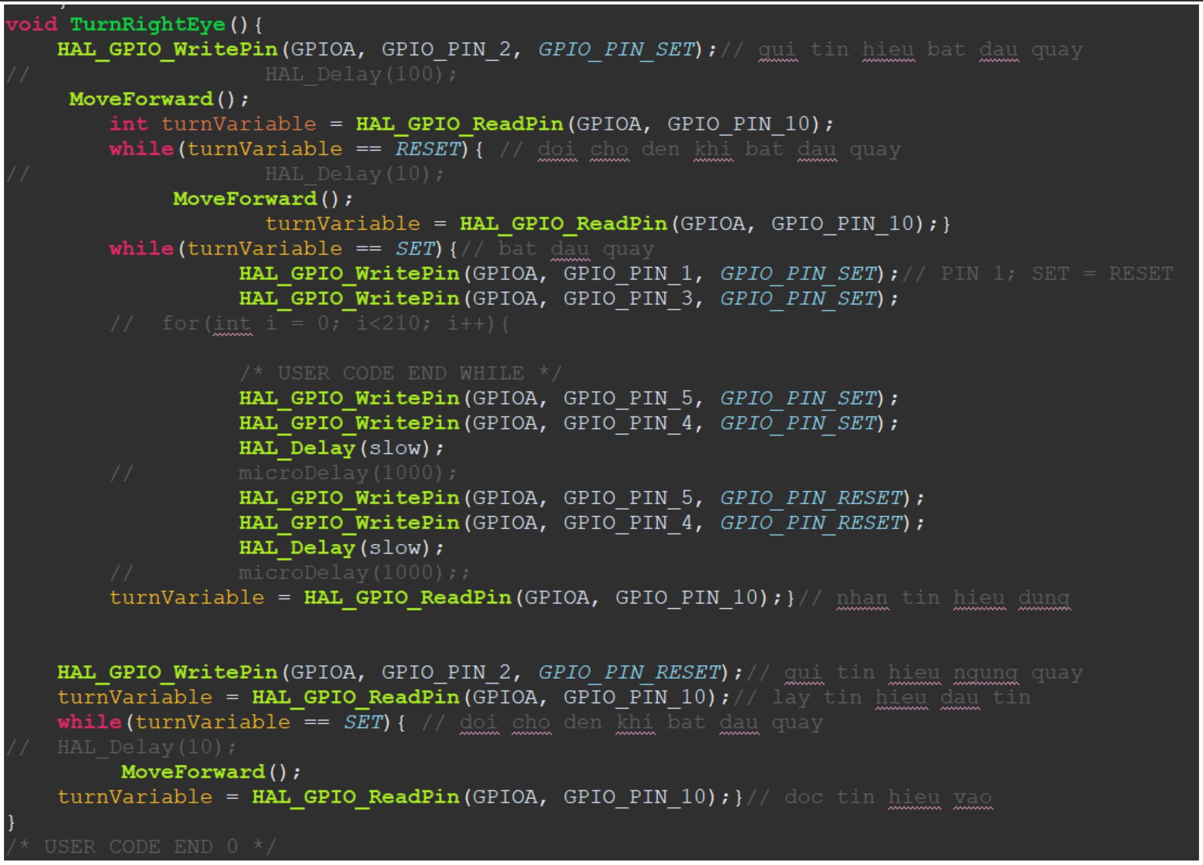
```c
void MoveForward(){

    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);// DAO LAI CHAN PIN 3

        /* USER CODE END WHILE */
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_Delay(slow);
//          microDelay(800);

        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
        HAL_Delay(slow);
//          microDelay(800);


}
void MoveShort(){
    for(int i = 0; i < 198; i++){
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);

        /* USER CODE END WHILE */
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_Delay(slow);
//          microDelay(800);

        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
        HAL_Delay(slow);
```



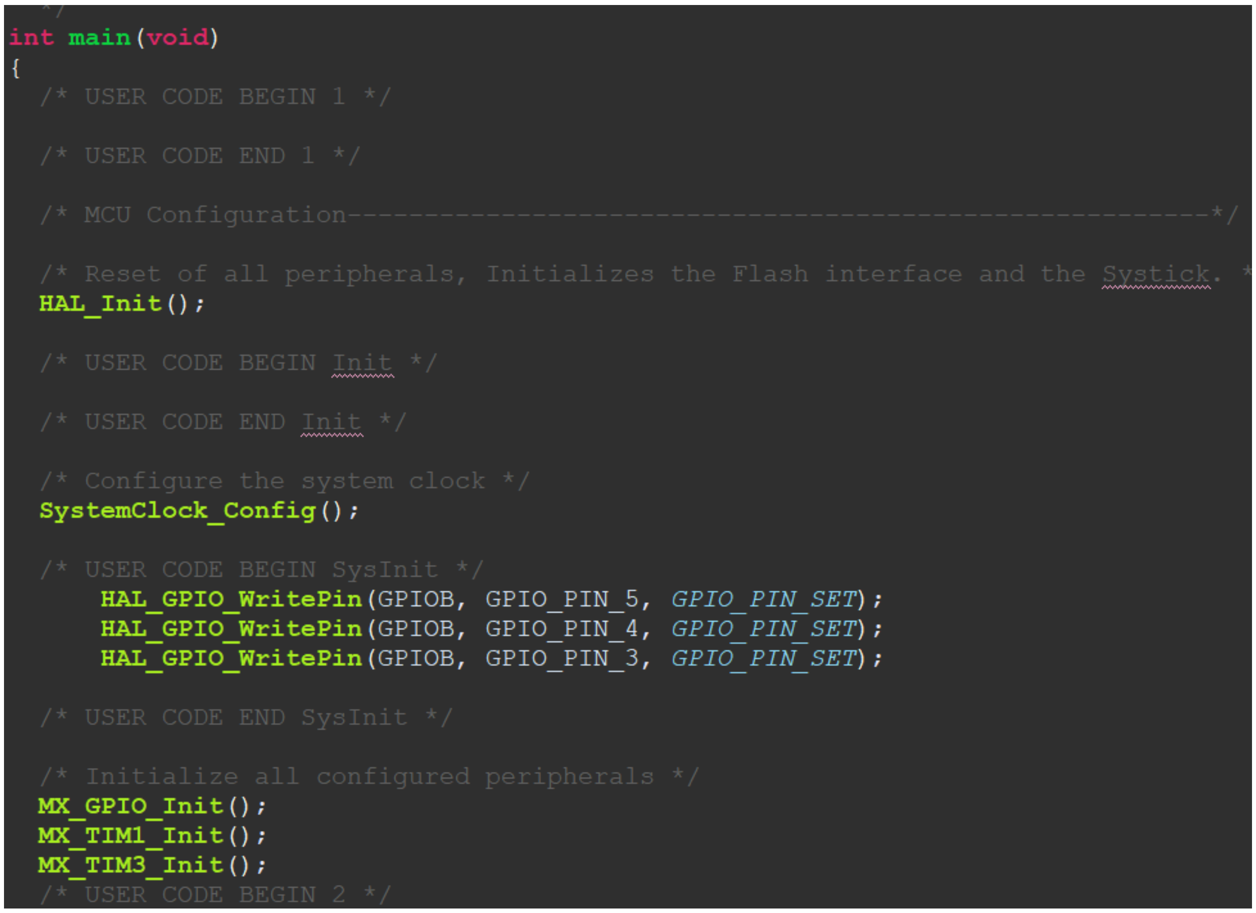
```c
void IncreaseSpeed(){
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);// PIN1 : RESET =
      HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);
      for(int x = 3; x < 1; x--){
          /* USER CODE END WHILE */
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
          HAL_Delay(x);
//            microDelay(800);

          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
          HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
          HAL_Delay(x);
//            microDelay(800);
      }
}
```



```
void TurnLeftEye(){
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET);// qui tin hieu bat dau quay
  //          HAL_Delay(100);
    MoveForward();
      int turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);
      while(turnVariable == RESET){ // doi cho den khi bat dau quay
  //          HAL_Delay(10);
        MoveForward();
            turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);}
      while(turnVariable == SET){// bat dau quay
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);// PIN 1; SET = RE
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_RESET);
      //  for(int i = 0; i<210; i++){

           /* USER CODE END WHILE */
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
           HAL_Delay(slow);
      //       microDelay(1000);
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
           HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
           HAL_Delay(slow);
      //       microDelay(1000);;
      turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);}// nhan tin hieu dung


    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);// qui tin hieu ngung quay
    turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);// lay tin hieu dau tin
    while(turnVariable == SET){ // doi cho den khi bat dau quay
  // HAL_Delay(10);
      MoveForward();
    turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);}// doc tin hieu vao
  }
```

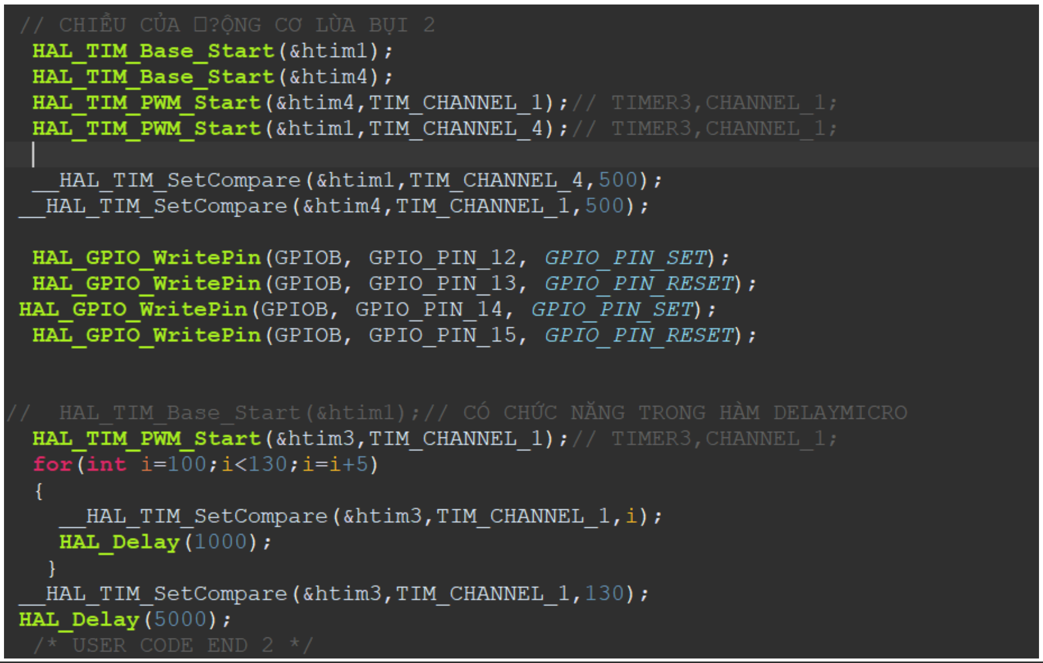


```
void TurnRightEye(){
  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_SET);// qui tin hieu bat dau quay
//          HAL_Delay(100);
  MoveForward();
    int turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);
    while(turnVariable == RESET){ // doi cho den khi bat dau quay
//          HAL_Delay(10);
      MoveForward();
          turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);}
    while(turnVariable == SET){// bat dau quay
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);// PIN 1; SET = RESET
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_3, GPIO_PIN_SET);
    //  for(int i = 0; i<210; i++){

         /* USER CODE END WHILE */
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_SET);
         HAL_Delay(slow);
    //       microDelay(1000);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
         HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, GPIO_PIN_RESET);
         HAL_Delay(slow);
    //       microDelay(1000);;
    turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);}// nhan tin hieu dung


  HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);// qui tin hieu ngung quay
  turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);// lay tin hieu dau tin
  while(turnVariable == SET){ // doi cho den khi bat dau quay
// HAL_Delay(10);
    MoveForward();
  turnVariable = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_10);}// doc tin hieu vao
}
/* USER CODE END 0 */
```

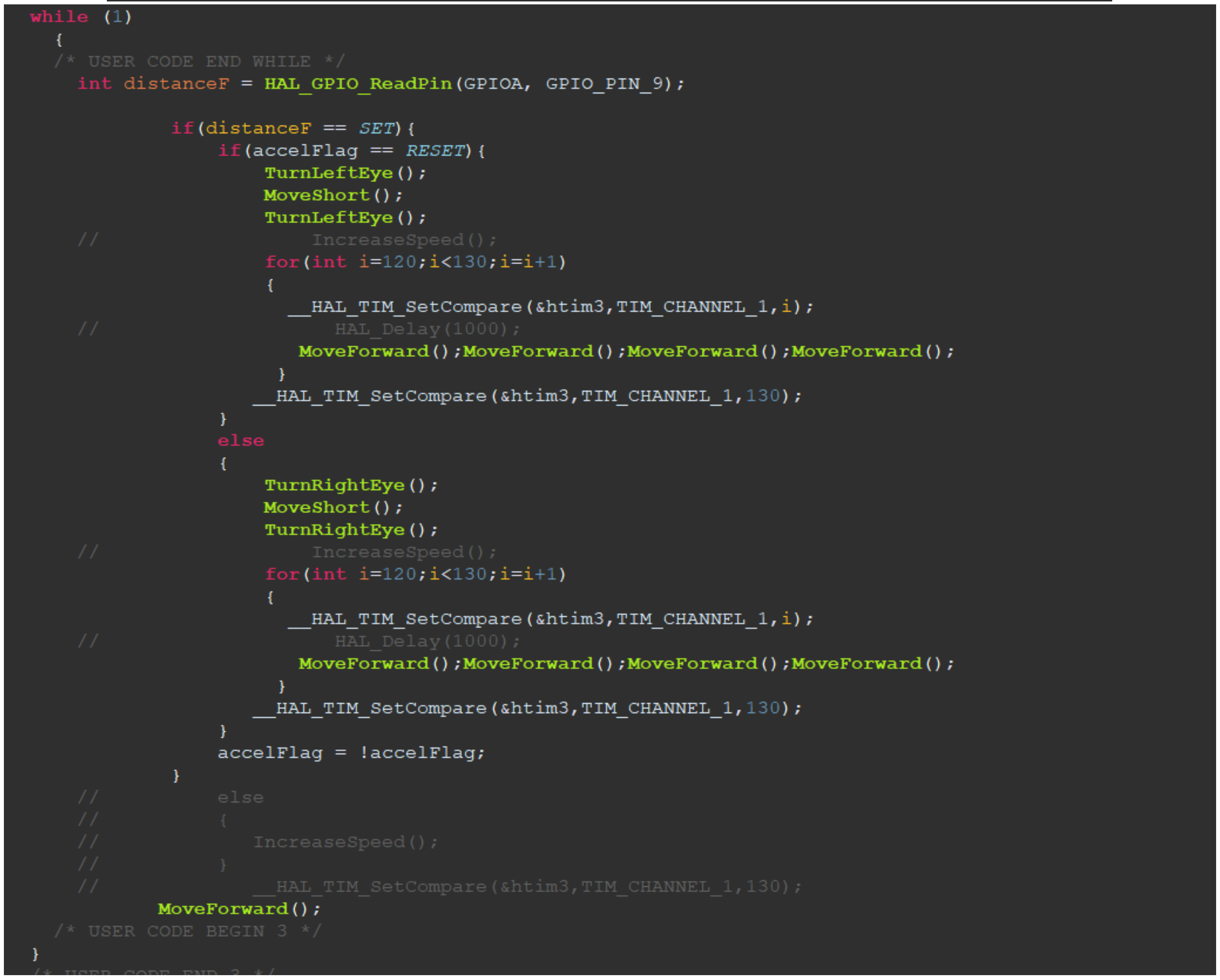


```
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. *
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, GPIO_PIN_SET);

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */
```

❖ **activate the vacuum cleaner motor**



```c
// CHIỀU CỦA □?ỘNG CƠ LÙA BỤI 2
 HAL_TIM_Base_Start(&htim1);
 HAL_TIM_Base_Start(&htim4);
 HAL_TIM_PWM_Start(&htim4,TIM_CHANNEL_1);// TIMER3,CHANNEL_1;
 HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_4);// TIMER3,CHANNEL_1;
 
 __HAL_TIM_SetCompare(&htim1,TIM_CHANNEL_4,500);
__HAL_TIM_SetCompare(&htim4,TIM_CHANNEL_1,500);

 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, GPIO_PIN_RESET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, GPIO_PIN_SET);
 HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, GPIO_PIN_RESET);


//  HAL_TIM_Base_Start(&htim1);// CÓ CHỨC NĂNG TRONG HÀM DELAYMICRO
 HAL_TIM_PWM_Start(&htim3,TIM_CHANNEL_1);// TIMER3,CHANNEL_1;
 for(int i=100;i<130;i=i+5)
 {
   __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,i);
   HAL_Delay(1000);
  }
__HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,130);
HAL_Delay(5000);
 /* USER CODE END 2 */
```



```c
while (1)
  {
  /* USER CODE END WHILE */
    int distanceF = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_9);

            if(distanceF == SET){
                if(accelFlag == RESET){
                    TurnLeftEye();
                    MoveShort();
                    TurnLeftEye();
    //                  IncreaseSpeed();
                    for(int i=120;i<130;i=i+1)
                    {
                      __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,i);
    //                  HAL_Delay(1000);
                       MoveForward();MoveForward();MoveForward();MoveForward();
                     }
                   __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,130);
                }
                else
                {
                    TurnRightEye();
                    MoveShort();
                    TurnRightEye();
    //                  IncreaseSpeed();
                    for(int i=120;i<130;i=i+1)
                    {
                      __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,i);
    //                  HAL_Delay(1000);
                       MoveForward();MoveForward();MoveForward();MoveForward();
                     }
                   __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,130);
                }
                accelFlag = !accelFlag;
            }
    //          else
    //          {
    //             IncreaseSpeed();
    //          }
    //             __HAL_TIM_SetCompare(&htim3,TIM_CHANNEL_1,130);
           MoveForward();
  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```

CODE IN ARDUINO

```
#include<math.h>
#include <MPU6050_tockn.h>
#include <Wire.h>
MPU6050 mpu6050(Wire);
//turn variable
#define Pinout  12//10stm32
#define Pinin   13//2stm32

// wall variable
//int distanceLeft = 3;//10
int distanceForward = 2;//9
//int distanceRight = 4;//8

long distance_L;
long distance_F;
long distance_R;

const int trigPin_left = 9;
const int echoPin_left = 10; // chan echo cua ultrasonic

const int trigPin_foward = 5;
const int echoPin_foward = 6; // chan echo cua ultrasonic

const int trigPin_righ = 7;
const int echoPin_righ = 8; // chan echo cua ultrasonic

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);

pinMode(Pinout, OUTPUT);
pinMode(Pinin, INPUT);

  pinMode(trigPin_left,OUTPUT);// CHAN PHAT TIN HIEU
  pinMode(echoPin_left, INPUT);// chan nhan tin hieu

  pinMode(trigPin_foward,OUTPUT); // trig lay tin hieu cho ultr
  pinMode(echoPin_foward, INPUT); // echo nhan lai tin hieu
  pinMode(trigPin_righ,OUTPUT);
  pinMode(echoPin_righ, INPUT);

  //pinMode(distanceLeft ,OUTPUT); // Step chân xung
  pinMode(distanceForward,OUTPUT); // Dir xác định chiều quay
//  pinMode(distanceRight,OUTPUT);
digitalWrite(Pinout, LOW);// cai dat ban dau
delay(10);
}
long microsecondsToInches(long microseconds)
{
  return microseconds/ 74 / 2;
}
long microsecondsToCentimeters(long microseconds)
{
  return microseconds/ 29 /2;
}
/////////////////////////////////////////////////////////////////////////////////////
////////////////
long Ultrasonic_left(long distance_L)
{
  long duration, inches;

digitalWrite(trigPin_left, LOW);
delayMicroseconds(2);

digitalWrite(trigPin_left, HIGH);
delayMicroseconds(10);

digitalWrite(trigPin_left, LOW);

duration = pulseIn(echoPin_left, HIGH);// DO DO RONG XUNG
inches = microsecondsToInches(duration); // ham chuyen doi inches
distance_L = microsecondsToCentimeters(duration);// ham chuyen doi sang distance_R

// delay(100);
return distance_L;
}
///////////////////////////////////////////// ULTRASONIC_FOWARD
///////////////////////////
long Ultrasonic_foward(long distance_F)
{
  long duration, inches;
```

```
digitalWrite(trigPin_foward, LOW);
delayMicroseconds(2);

digitalWrite(trigPin_foward, HIGH);
delayMicroseconds(10);

digitalWrite(trigPin_foward, LOW);

duration = pulseIn(echoPin_foward, HIGH);// DO DO RONG XUNG
inches = microsecondsToInches(duration); // ham chuyen doi inches
distance_F = microsecondsToCentimeters(duration);// ham chuyen doi sang distance_R

// delay(100);
return distance_F;
}
/////////////////////////////////////////////////////////////////////////////////////
//////////////
long Ultrasonic_righ(long distance_R)
{
  long duration, inches;

digitalWrite(trigPin_righ, LOW);
delayMicroseconds(2);

digitalWrite(trigPin_righ, HIGH);
delayMicroseconds(10);

digitalWrite(trigPin_righ, LOW);

duration = pulseIn(echoPin_righ, HIGH);// DO DO RONG XUNG
inches = microsecondsToInches(duration); // ham chuyen doi inches
distance_R = microsecondsToCentimeters(duration);// ham chuyen doi sang distance_R

// delay(1);
return distance_R;
}
////////////////////////////////////////////////////////////////////////
void Turn(){
digitalWrite(Pinout, LOW);
int firstAngle;
float AngleZ ;
for(int i = 0; i< 500;i++){
   mpu6050.update();
AngleZ = mpu6050.getAngleZ();
Serial.println(AngleZ);
}
Serial.print("AngleZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
ZZZZZZZZZZZZZZZZZZZZZZZZZZ: ");
 mpu6050.update();
Serial.println(AngleZ);

int count = 1;
if ( count = 1) { firstAngle = AngleZ; count=2;}

while ( abs(AngleZ - firstAngle) <= 180){
digitalWrite(Pinout , HIGH); // gui tin hieu ra de quay chan 12arduino//10stm32 vs
chan 10
AngleZ = mpu6050.getAngleZ();
 mpu6050.update();
Serial.println(AngleZ);
Serial.print("high ");

}
digitalWrite(Pinout, LOW);// gui tin hieu cho pin 10stm ngung quay
int turn = digitalRead(Pinin);//chan 2stm gui chan 11arduino nhan tin hieu
Serial.print("turn ");
Serial.println(turn);

if(turn == HIGH)
{
turn = digitalRead(Pinin);
Serial.print("turn ");
Serial.println(turn);
delay(10);
}
}

void loop()
{
int turn = digitalRead(Pinin);//chan 2stm gui chan 11arduino nhan tin hieu
Serial.print("turn ");
Serial.println(turn);

if(turn == HIGH)
{
Turn();
}
  int x = 7;
//  distance_L = Ultrasonic_left(distance_L);
  distance_F = Ultrasonic_foward(distance_F);
 // distance_R = Ultrasonic_righ(distance_R);

        if(distance_F <= x){ digitalWrite(distanceForward, HIGH);
        }
        if(distance_F >  x){ digitalWrite(distanceForward, LOW);
}
```