KING MONGKUT'S INSTITUTE OF TECHNOLOGY LATKRABANG
SCHOOL OF ENGINEERING
GROUP 4

<u>01461318 – MICROPROCESSOR AND INTERFACE</u>

FINAL PROJECT
HIGH SPEED LINE FOLLOWING PROJECT

INSTRUCTED BY: DR. POOM KONGHUAYROB & SUSHISH BARAL

NAME: CHISSANUPONG, PYAE, PICHAPA
ID NUMBER: 65011292, 65011497, 65011467
GROUP NO: 4

TABLE OF CONTENTS

Abstract

Line following robots are autonomous robotic systems designed to navigate along a predetermined path by following a line on the ground. These robots are widely used in various applications, including industrial automation, transportation systems, and educational platforms. The objective of a line following robot is to accurately track and follow a contrasting line while maintaining a consistent speed and direction.

Several challenges arise in the development and implementation of line following robots. These challenges include accurate line detection under varying lighting conditions, handling intersections and curves, dealing with line discontinuities, and maintaining stability and speed control. Researchers and engineers continue to explore innovative solutions to improve the robustness, adaptability, and efficiency of line following robots.

A high-speed line following robot that leverages state-of-the-art technology to efficiently track and follow lines with remarkable speed and precision.
This line-following robot's main goal is to quickly travel predetermined courses while precisely following a line. This will allow it to go through the course given. To achieve its high-speed line following capabilities, the robot combines sensors, and feedback control algorithms.

Introduction

The robot has a PID control system to ensure accurate and quick movement. The closed-loop feedback mechanisms are used in the control system to continuously track the position of the line with respect to the robot's sensors, allowing for quick course modifications and precise tracking. The microprocessor in use is a STM32 F411RE. A 16-bit processor uses C programming to create system

The STM32 F411RE microcontroller is used as the central processing unit and PID (Proportional-Integral-Derivative) control coding is used for precise line tracking and control in this conception and implementation of a line-following robot. With its extensive features and processing power, the STM32 F411RE microcontroller makes a perfect platform for creating complex robotic systems.

Throughout this report, we will explain the design considerations, hardware setup, and software development processes involved in building the line following robot. We will discuss the integration of infrared sensors for line detection. Additionally, we will explore the implementation of PID control coding on the STM32 F411RE microcontroller, focusing on the tuning of PID parameters for optimal performance.

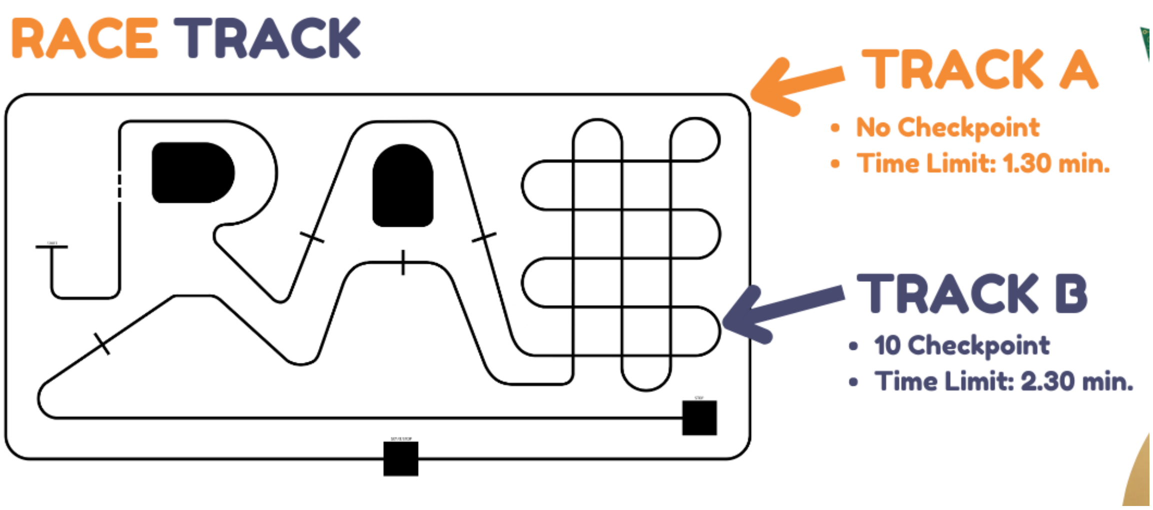
By the end of this report, we aim to demonstrate the successful construction and operation of a line following robot using the STM32 F411RE microcontroller and PID control coding. The insights gained from this project will contribute to the advancement of autonomous navigation systems and provide a foundation for future developments in robotics and automation.

Purpose

1) To learn about sensor normalization and feedback control like PID control. Build a strong foundation on how to stabilize the robot's movement.
2) Learn how to draw and read the schematic for the electrical components.
3) Learn to design a PCB and use it as a chassis.
4) For the robot to be able to follow the line in the map given within the lime limit.

Scope of the project

1) Make a high-speed line following robot. Using PID feedback control algorithm. Make the chassis and assemble all the components for the robot.
2) Use line sensing by implementing the infrared sensors for the robot to determine its position.
3) Use the STMF411RE as the microprocessor for the software development.
4) Motor control is the process of controlling the speed and direction of DC motors.
5) Design the PCB for placing the components and routing the connections to create a compact efficient design.
6) There will be 2 tracks, Tracks "A" and "B". Track A will have a time limit of 1.30 minutes. And track B will have a 2.30 minutes time limit. If the robot goes out of track, you can start at the latest point for track B. But for track A you have to start at the starting point.
7) The size of the robot is limited to 230x230 millimeters square. The line width is 19 millimeters. Robot must remain tethered to the power supply while racing.



Background research

Sensor normalization refers to the process of transforming sensor readings into a standardized range or format that can be easily interpreted and used for control purposes. In the context of a line following robot, sensor normalization plays a crucial role in ensuring accurate line detection and tracking.

- Calibration: Calibrating sensors involves determining their response characteristics and establishing a relationship between the sensor readings and the physical quantity being measured. This process helps eliminate sensor bias and provides consistent and reliable measurements.

- Range Mapping: Once the sensors are calibrated, range mapping is performed to scale the sensor readings to a desired range. This step ensures that the sensor outputs are within a defined and manageable range, facilitating easier interpretation and processing.

- Normalization Techniques: Normalization techniques, such as linear scaling or z-score normalization, are employed to map the sensor readings to a common range. This allows for consistent comparison and integration of sensor data during the control process.

Understanding Feedback Control: PID Control:

Feedback control is a fundamental concept in robotics and control systems, enabling autonomous systems to continuously adjust their actions based on sensed feedback. The PID (Proportional-Integral-Derivative) control algorithm is a widely used method for feedback control due to its simplicity and effectiveness.

To develop a strong foundation in PID control, it is important to grasp the following concepts:

- Proportional Control (P): The proportional term produces a control action that is proportional to the error between the desired reference value (setpoint) and the measured value (feedback). This term helps achieve stability by providing a control action that scales with the error.

- Integral Control (I): The integral term calculates a control action based on the accumulation of past errors. It helps address steady-state errors and improves the system's ability to reach and maintain the setpoint.

- Derivative Control (D): The derivative term calculates a control action based on the rate of change of the error. It helps improve system response and stability by anticipating future changes and minimizing overshoot or oscillations.

- Tuning PID Parameters: Tuning PID parameters involves adjusting the relative weights of the proportional, integral, and derivative terms to achieve desired system performance. This process is typically performed iteratively, analyzing the system's response and adjusting the parameters accordingly.

- PID Implementation: Implementing PID control involves computing the control action based on the PID equation, considering the current error, the accumulated error, and the rate of change of the error. The control action is then applied to actuator(s) to adjust the system's behavior.

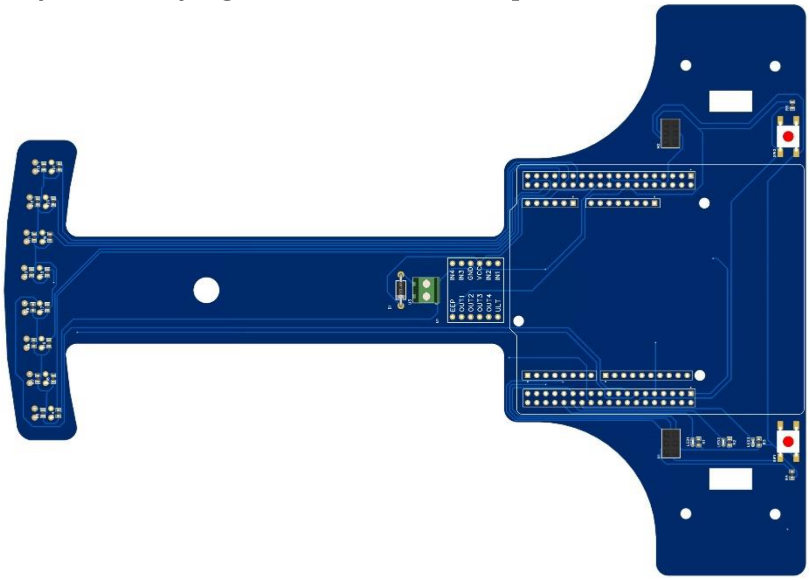
The PID controller's output is calculated as the sum of the proportional, integral, and derivative terms: Output = Kp * Error + Ki * Integral of Error + Kd * Derivative of Error

By gaining a solid understanding of sensor normalization and feedback control, specifically PID control, you will be equipped with the necessary knowledge and skills to stabilize the movement of a line following robot. These foundations will enable you to effectively utilize sensor data, apply appropriate control algorithms, and fine-tune the robot's behavior to achieve accurate and stable line tracking.

Methodology

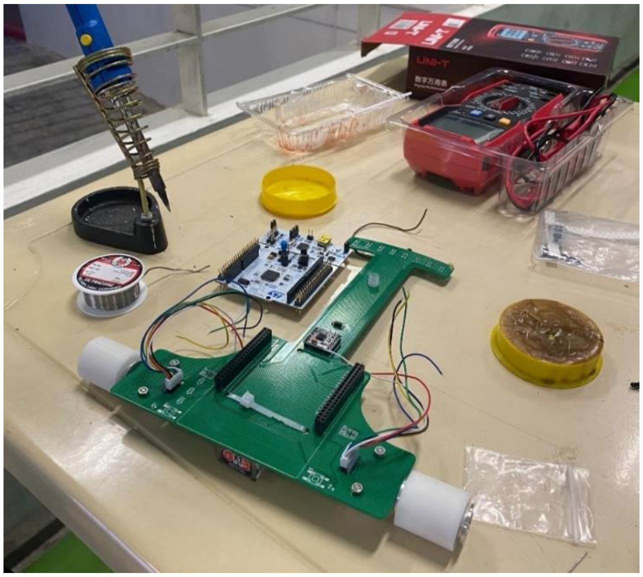
- Build the schematic design for the electric components of the robot. Building a schematic design involves creating a visual representation of the electrical and electronic components, connections, and their relationships within a system or circuit. Remember to consult datasheets, reference designs, and relevant resources while creating the schematic design to ensure the correct placement and connections of components. It's also beneficial to follow standard schematic design practices and conventions to enhance readability and ease of understanding for others who may review or use your design.

- Design the PCB board to use it as the chassis. For this design the sensors are 10mm apart from each other for there to be less error. IR sensors is for identifying the position of the line with in the robot’s sensors range by detecting the infrared light reflected off the surface. All of the IR sensors are placed with the same radius from the midpoint between motors. The IR sensors are placed furthest away from the microprocessor and motors for more stability in identifying the line. The LED is placed in front of the sensors so that the



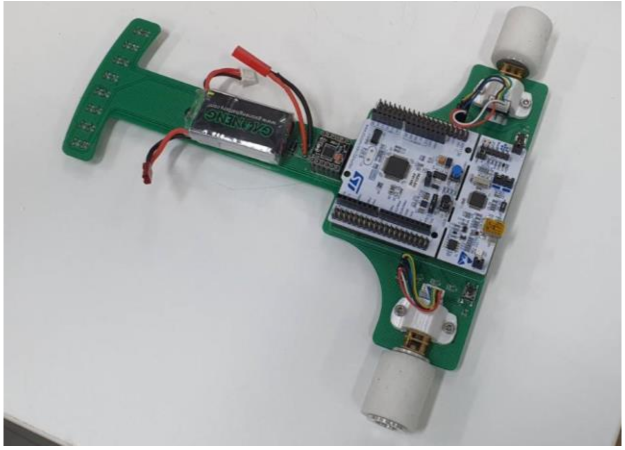
light can bounce back to the sensors. The robot is designed to weigh as less as possible for mobility purposes.

- After solder all the components on the PCB board we need to check if there is any disconnectivity on the PCB board after soldering by using a multimeter. Otherwise, the robot can’t function.



- Check if the IR sensors are working and are able to track the line. By writing the code to show the IR sensor values and monitor it. Implement sensor normalization and calibrating the sensors.

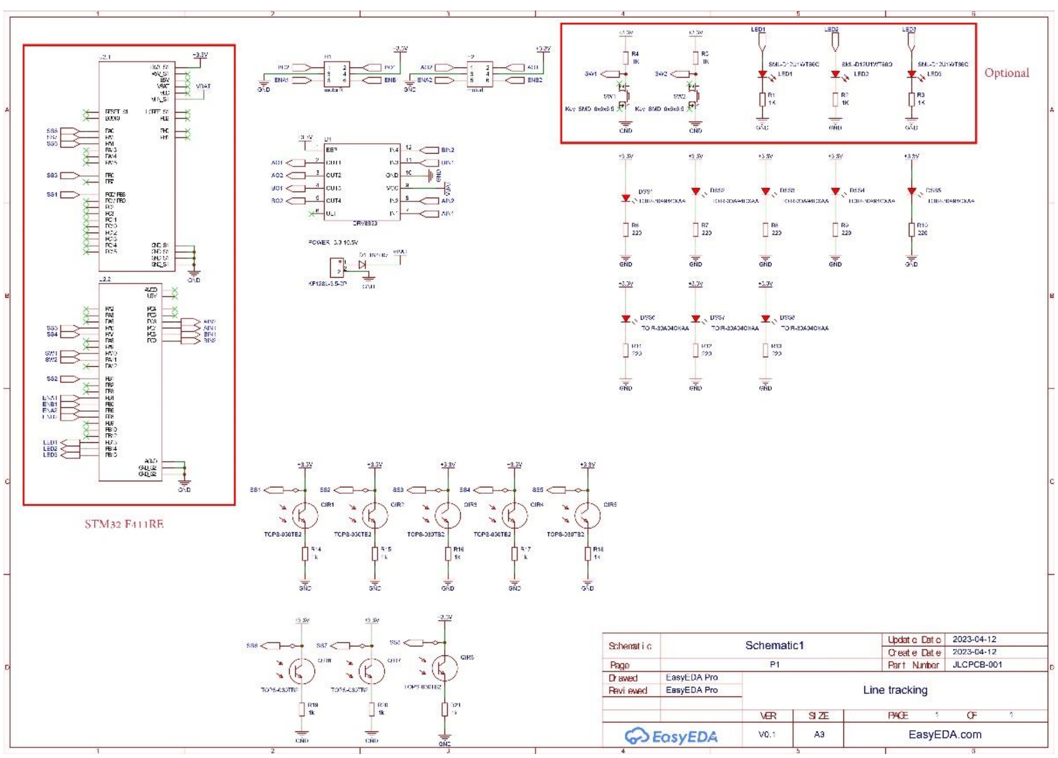
- Test for the robot stability write the logic for the robot’s movement. And find the appropriate power output for the motors. Implement feedback control, PID control. Finalize the code and tune the PID variables by monitoring the robot’s movement on the course given multiple times and tune it for the best results. The PID controller's output is calculated as the sum of the proportional, integral, and derivative terms:
$$Output = Kp * Error + Ki * Integral\ of\ Error + Kd * Derivative\ of\ Error$$



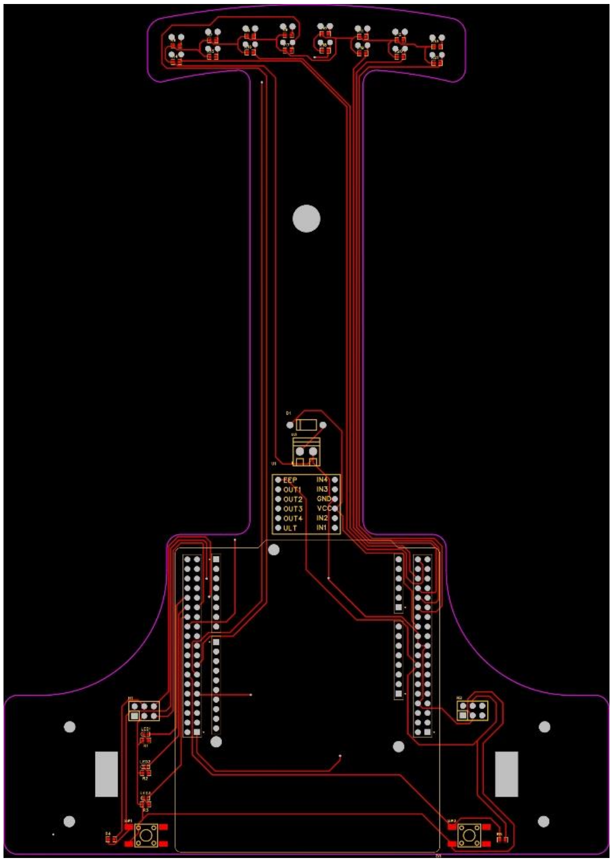
Components List

Name	Quantity
STM32 F411RE	1
Wheels	2
DC Motor	2
PCB board	1
Phototransistor TOPS-030TB2	8
Led TOIR-30A94CXAA	8
Resistor 1KOhms	13
Resistor 220 Ohms	8
Diode 1N4007	1
Motor Driver drv8833	1
0402 RGB led	3
Switch Button	2

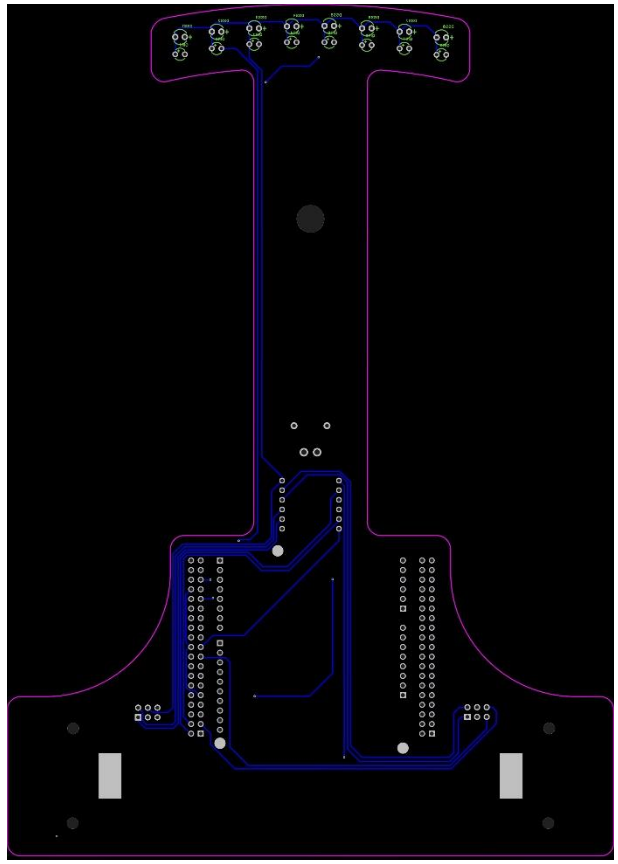
Schematic



PCB board design

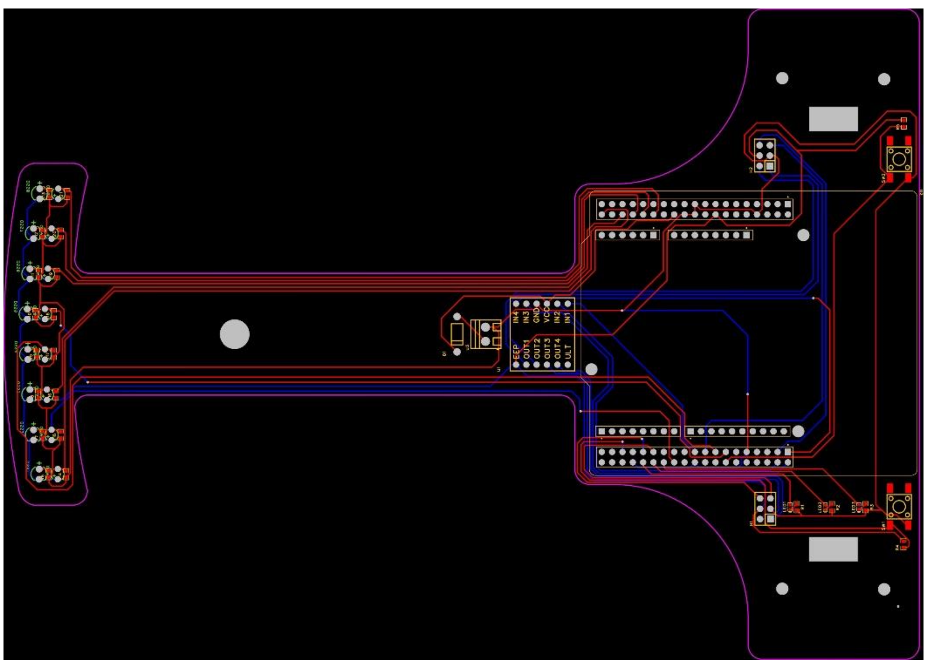


Front



Back

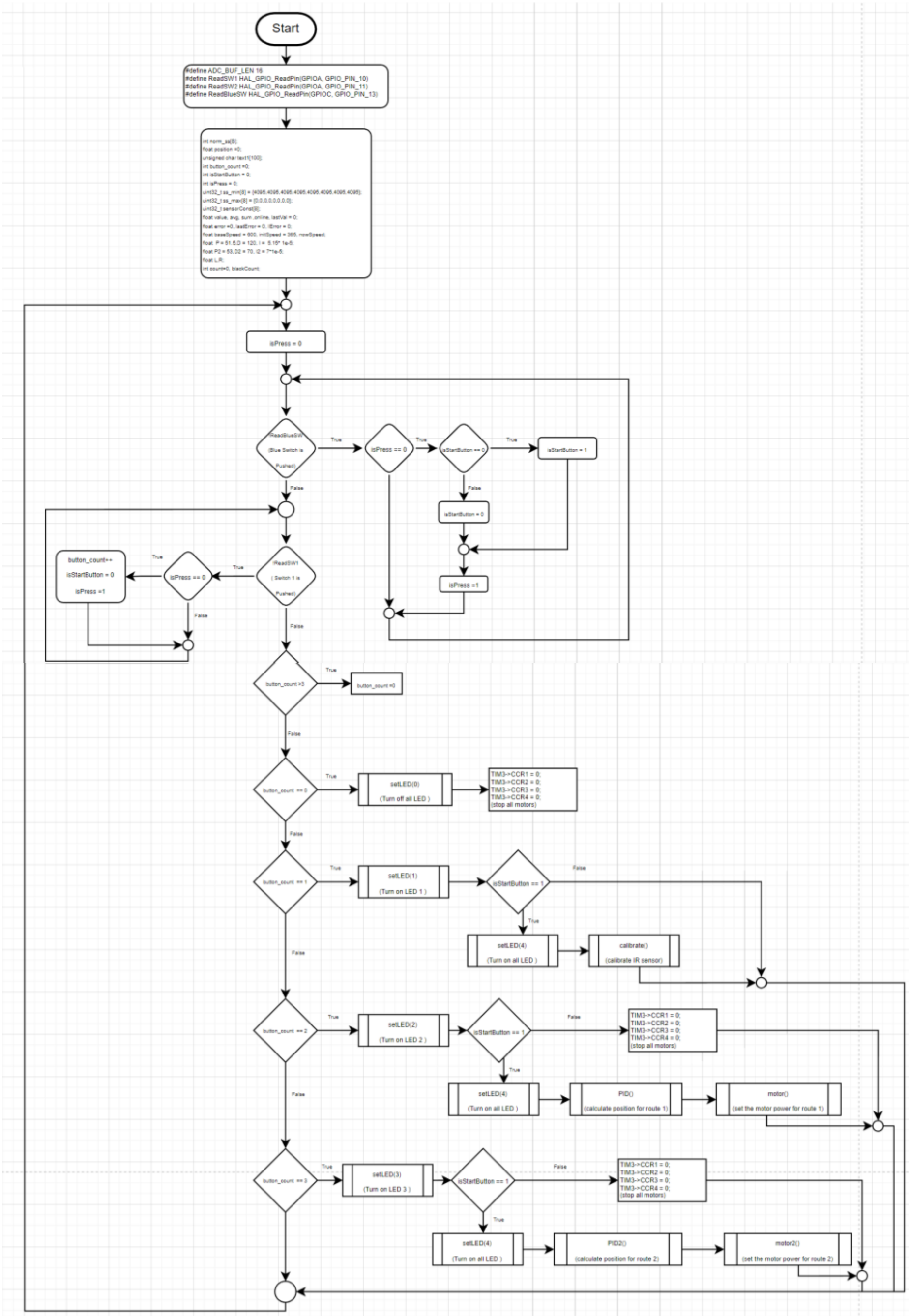
Overall

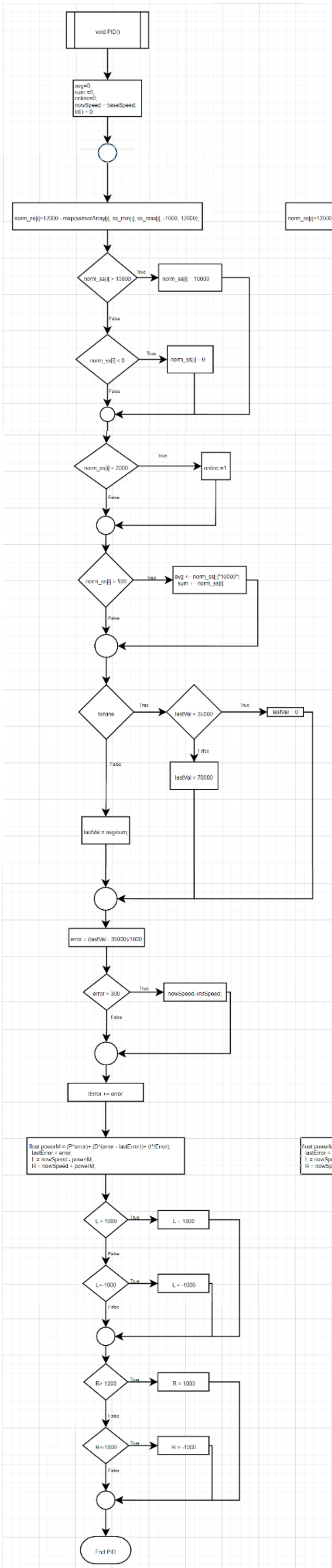


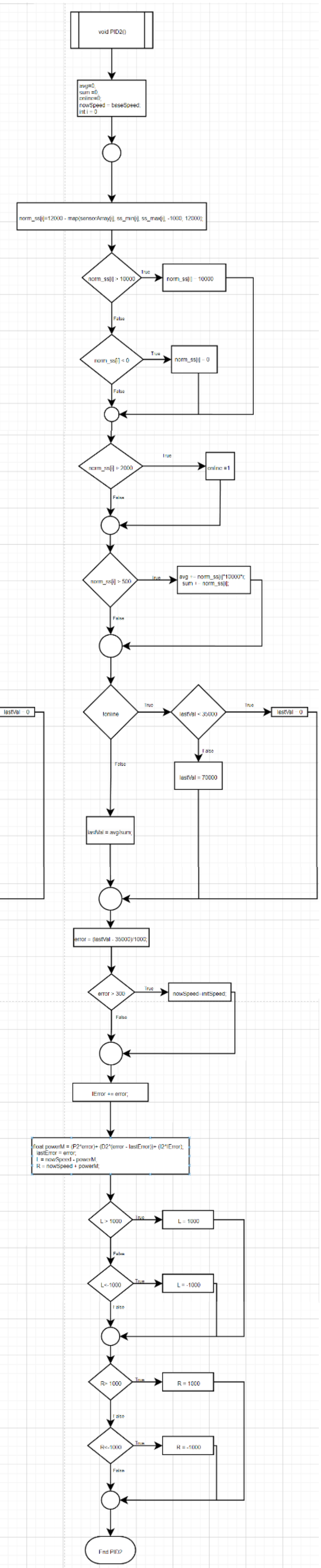
PID control and coding

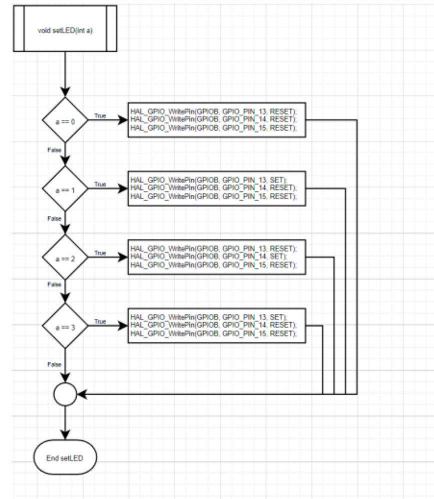
[PID code]

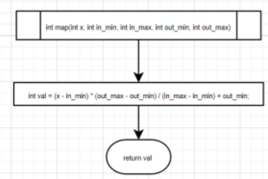
Flowchart

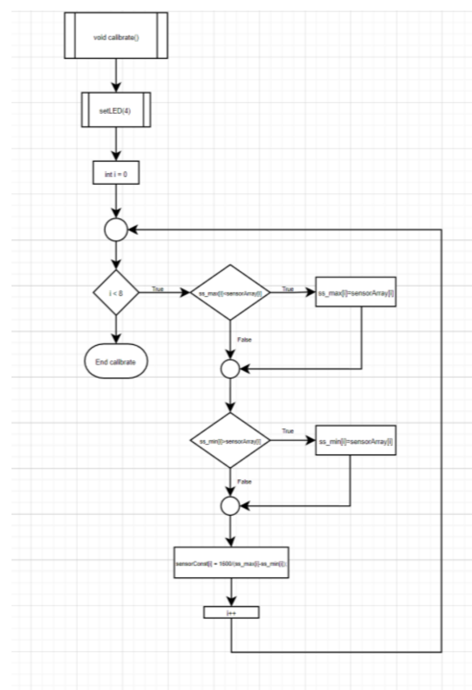


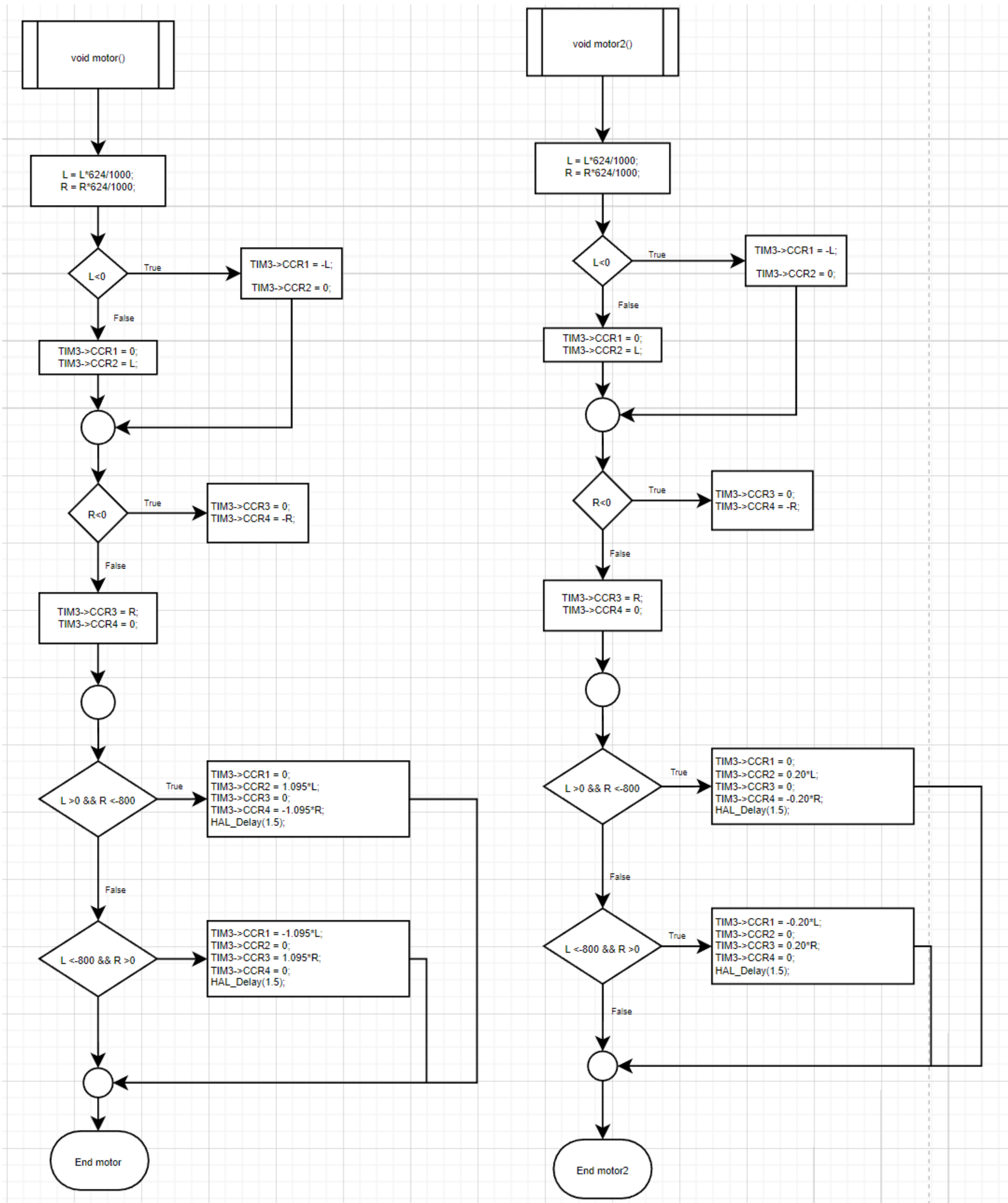

void PID()
avg=0;
sum =0;
online=0;
nowSpeed = baseSpeed;
int i = 0
norm_ss[i]=12000 - map(sensorArray[i], ss_min[i], ss_max[i], -1000, 12000);
norm_ss[i] > 12000
True
norm_ss[i] = 10000
False
norm_ss[i] < 0
True
norm_ss[i] = 0
False
norm_ss[i] > 2000
True
online =1
False
norm_ss[i] > 500
True
avg += norm_ss[i]*10000*i;
sum += norm_ss[i];
False
!online
True
lastVal < 35000
True
lastVal = 0
False
lastVal = 70000
False
lastVal = avg/sum;
error = (lastVal - 35000)/1000
error > 300
True
nowSpeed=initSpeed;
False
IError += error;
False
L > 1000
True
L = 1000
False
L<-1000
True
L = -1000
False
R> 1000
True
R = 1000
False
R<-1000
True
R = -1000
False
End PID


void PID2()
avg=0;
sum =0;
online=0;
nowSpeed = baseSpeed;
int i = 0
norm_ss[i]=12000 - map(sensorArray[i], ss_min[i], ss_max[i], -1000, 12000);
norm_ss[i] > 10000
True
norm_ss[i] = 10000
False
norm_ss[i] < 0
True
norm_ss[i] = 0
False
norm_ss[i] > 2000
True
online =1
False
norm_ss[i] > 500
True
avg += norm_ss[i]*10000*i;
sum += norm_ss[i];
False
!online
True
lastVal < 35000
True
lastVal = 0
False
lastVal = 70000
False
lastVal = avg/sum;
error = (lastVal - 35000)/1000
error > 300
True
nowSpeed=initSpeed;
False
IError += error;
L > 1000
True
L = 1000
False
L<-1000
True
L = -1000
False
R> 1000
True
R = 1000
False
R<-1000
True
R = -1000
False
End PID2


void setLED(int a)
a == 0
True
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, RESET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, RESET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, RESET);
False
a == 1
True
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, SET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, RESET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, RESET);
False
a == 2
True
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, RESET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, SET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, RESET);
False
a == 3
True
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, SET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, RESET);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, RESET);
False
End setLED


int map(int x, int in_min, int in_max, int out_min, int out_max)
int val = (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
return val


void calibrate()
setLED(4)
int i = 0
i < 8
True
ss_max[i]<sensorArray[i]
True
ss_max[i]=sensorArray[i]
False
End calibrate
ss_min[i]>sensorArray[i]
True
ss_min[i]=sensorArray[i]
False
i++


void motor()
L = L*624/1000;
R = R*624/1000;
L<0
True
TIM3->CCR1 = -L;
TIM3->CCR2 = 0;
False
TIM3->CCR1 = 0;
TIM3->CCR2 = L;
R<0
True
TIM3->CCR3 = 0;
TIM3->CCR4 = -R;
False
TIM3->CCR3 = R;
TIM3->CCR4 = 0;
L >0 && R <-800
True
TIM3->CCR1 = 0;
TIM3->CCR2 = 1.095*L;
TIM3->CCR3 = 0;
TIM3->CCR4 = -1.095*R;
HAL_Delay(1.5);
False
L <-800 && R >0
TIM3->CCR1 = -1.095*L;
TIM3->CCR2 = 0;
TIM3->CCR3 = 1.095*R;
TIM3->CCR4 = 0;
HAL_Delay(1.5);
End motor
void motor2()
L = L*624/1000;
R = R*624/1000;
L<0
True
TIM3->CCR1 = -L;
TIM3->CCR2 = 0;
False
TIM3->CCR1 = 0;
TIM3->CCR2 = L;
R<0
True
TIM3->CCR3 = 0;
TIM3->CCR4 = -R;
False
TIM3->CCR3 = R;
TIM3->CCR4 = 0;
L >0 && R <-800
True
TIM3->CCR1 = 0;
TIM3->CCR2 = 0.20*L;
TIM3->CCR3 = 0;
TIM3->CCR4 = -0.20*R;
HAL_Delay(1.5);
False
L <-800 && R >0
True
TIM3->CCR1 = -0.20*L;
TIM3->CCR2 = 0;
TIM3->CCR3 = 0.20*R;
TIM3->CCR4 = 0;
HAL_Delay(1.5);
False
End motor2

Member

Mr. CHISSANUPONG SAENGSINT 65011292 (33.33%)

Soldering, Coding, Tuning PID, Report, Video

Chissanupong.

Signature:

Miss PICHAPA HANSATIT 65011467 (33.33%)

Soldering, Tuning PID, Report, Poster

Signature:

Mr. PYAEHTOO KHANT 65011497 (33.33%)

PCB design, Soldering, Tuning PID, Report, Video

Pyae

Signature:

Link for the video:

Video Link

Link for poster:

Poster

Reference

- Danny Jost (2019). What is IR sensor. Retrieved from fierce electronics: https://www.fierceelectronics.com/sensors/what-ir-sensor
- Electrical4U (2020). PID control. Retrieved from electrical4u: https://www.electrical4u.com/pid-control/
- JIMBLOM- How to read a schematic. Retrieved from earn.sparkfun: https://learn.sparkfun.com/tutorials/how-to-read-a-schematic/all
- PID controller & Theory Explain. Retrieved from NI engineer ambitiously: https://www.ni.com/en-th/shop/labview/pid-theory-explained.html