d

**VIETNAM NATIONAL UNIVERSITY, HANOI**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

[](http://test.uet.vnu.vn/)

STUDENT FINAL PROJECT

SMART ROBOT SYSTEM

2017 - 2018

**LINE FOLLOWER CAR ROBOT IN GETTING OUT MAZE AND FINDING OPTIMAL PATH SUPPORT INDOOR LOCALIZATION AND NAVIGATION**

Hoang The Hop QH-2015-I/CQ-ĐA

Tran Xuan Tuyen QH-2015-I/CQ-ĐA

Doan Manh Duy QH-2015-I/CQ-ĐA

**Research advisor:** **DR. NGUYEN THI THANH VAN**

**HA NOI - 2016**

**TABLE OF CONTENTS**

[Table of contents 1](#_Toc509845041)

[List of figures 2](#_Toc509845042)

[Objective 3](#_Toc509845043)

[I. System components 4](#_Toc509845044)

[1.1. Components 4](#_Toc509845045)

* + 1. Arduino Uno 4
    2. MPU 6050 GY 521 5
    3. Module L298N 5
    4. Channel Line Tracking Sensor Module 6

[1.2. Connection and operating principle 8](#_Toc509845046)

[II. Proposed method 9](#_Toc509845047)

[2.1. PID controller 9](#_Toc509845048)

* + 1. PID controller introduction 9
    2. PID tuning 9
    3. Applying PID in our line following robot 10

[2.2. The Kalman Filter in our line following robot 12](#_Toc509845049)

* 1. Applying PID in our line following robot 13
  2. Android application for user control 17

III. Result 17

Reference 19

# List of figures

[*Figure 1. Block diagram of the system 4*](#_Toc509480774)

[*Figure 2. MPU 6050 5*](#_Toc509480775)

[*Figure 3. Module L298N 5*](file:///E:\NCKH_2017\IPS_in_fire\BarometerArticle\BarometerReport.docx#_Toc509480776)

[*Figure 4. PID control schematic 8*](file:///E:\NCKH_2017\IPS_in_fire\BarometerArticle\BarometerReport.docx#_Toc509480777)

[*Figure 5.*Effects of increasing a parameter independently 9](#_Toc509480778)

[*Figure 6. PID parameter tuning* 10](#_Toc509480779)

[*Figure 7. System controller* 11](#_Toc509480780)

[*Figure 8.* Maze example 13](#_Toc509480781)

[*Figure 9. Algorithm Flow Chart* 14](#_Toc509480782)

[*Figure 10.* Matlab simulation results*.* 15](#_Toc509480783)

**List of tables**

[*Table 1.* Connecting BFD with adruino *7*](#_Toc446083073)

[*Table 2.* Connecting L298N with adruino 8](#_Toc446083074)

[*Table 3.* Connectiong MPU6050 with adruino 8](#_Toc446083074)

[*Table 4.* Actions at an intersection 13](#_Toc446083074)

[*Table 5.* List of triad reduction 13](#_Toc446083074)

[*Table 6.* Mapping sensor value with corresponse error 16](#_Toc446083074)

# Objective

Build a smart robot system have an ability to follow the black line on the road. In addition, the proposal robot also can find the a correct path in an any available maze. After that, in the next implementation, our robot can automatically remember which way to follow without waste time to find the best journey again.

# System components

The system including three main components: Kit Arduino Uno, Module L298N, 5 channel line tracking sensor module, MPU 6050 – GY521.

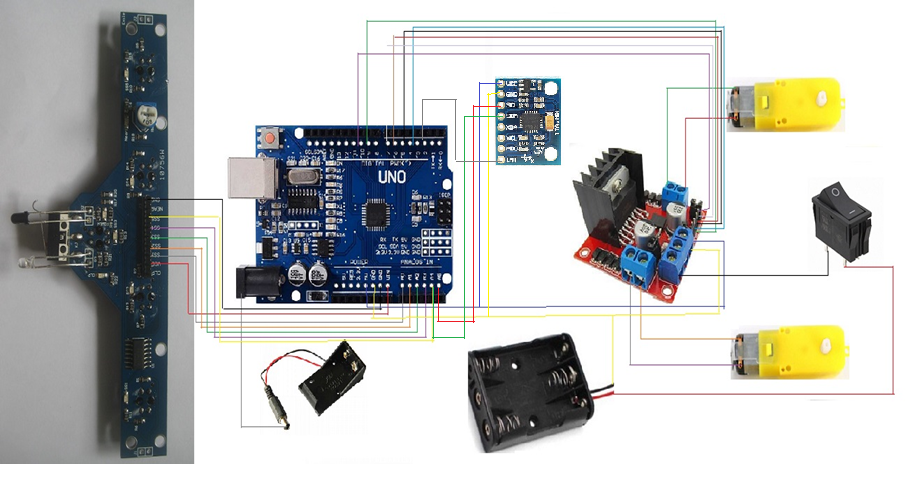


Figure 1. Block diagram of the system

### Components

#### Arduino Uno

The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button.

#### MPU 6050 GY521

.The MPU-6050 accelerometer sensor GY-521 integrates 3-axis accelerator + 3-axis gyroscope to control the balance or direction of motion for robots, aircraft, drone, gaming console, balance for camera / camera, recognition of fall, vibration, wax, ...

Figure 2. MPU 6050

Technical Specifications

• Chip: MPU-6050 (16bit ADC, 16bit data out)

• Gyroscapes in range: +/- 250 500 1000 2000 degree / sec

• Acceleration value: +/- 2g, +/- 4g, +/- 8g, +/- 16g

• Communication: I2C

• Power supply: 3V - 5V (DC)

#### Module L298N

|  |  |
| --- | --- |
| Figure 3. Module L298N |  |

- The MPU-6050 accelerometer sensor GY-521 integrates 3-axis accelerator + 3-axis gyroscope to control the balance or direction of motion for robots, aircraft, drone, gaming console, balance for camera / camera, recognition of fall, vibration, wax, ...

Technical Specifications

* Chip: MPU-6050 (16bit ADC, 16bit data out)
* Gyroscapes in range: +/- 250 500 1000 2000 degree / sec
* Acceleration value: +/- 2g, +/- 4g, +/- 8g, +/- 16g
* Communication: I2C
* Power supply: 3V - 5V (DC)

### Channel Line Tracking Sensor Module (BFD-1000)

5 Channel Line Tracking Sensor Module (BFD-1000) is a sensor board designed for use with line follower robots. This module has been sufficient to meet the day-to-day task of tracking, but also with the infrared distance sensor and touch detection sensor, the board makes your robot design able to adapt to the situation easily.

### Connection and operating principle

|  |  |
| --- | --- |
| BFD-1000 | Arduino |
| VCC | Vin |
| S1 | A0 |
| S2 | A1 |
| S3 | A2 |
| S4 | A3 |
| S5 | D11 |
| GND | GND |

Table 1: Connecting BFD with adruino

|  |  |  |  |
| --- | --- | --- | --- |
| L298N | Arduino | Motor | Switch |
| 12V |  |  | Positive |
| GND | GND |  | Negative |
| 5V | 5V |  |  |
| EnbA | D9 |  |  |
| IN1 | D4 |  |  |
| IN2 | D5 |  |  |
| IN3 | D6 |  |  |
| IN4 | D7 |  |  |
| EnbB | D10 |  |  |
| OutA |  | Left |  |
| OutB |  | Right |  |

**Table 2: Connecting L298N with adruino**

|  |  |
| --- | --- |
| MPU–6050 | Arduino |
| VCC | 5V |
| GND | GND |
| SCL | A4 |
| SDA | A5 |
| INT | D2 |

**Table 3: Connectiong MPU6050 with adruino**

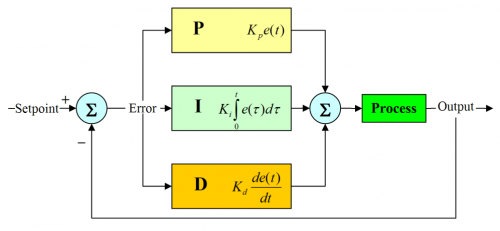
# Proposed method

### PID controller

PID is an important controller unit integrated in many closed-loop feedback system. The three term P, I, D stand for proportional, integral and derivative respectively. The implementation of this controller is took place in a range of product in academic as well as industry environment.

#### PID controller introduction

The fundamental idea of PID controller algorithm is to calculate the “PID value” based on the error behaviour. The error value is considered as the difference betweenameasured process and desired set point.



**Figure 4: PID control schematic**

#### PID tuning

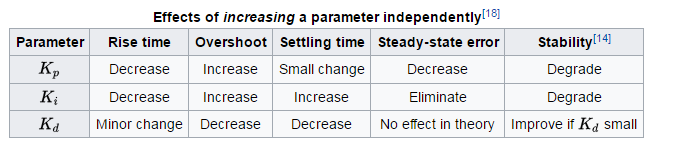
The calculation of PID controller involves the estimate three different parameters: the proportional, the integral and derivative values, denotes P, I and D.

The proportion term, Kp depends on the present error. A large Kp value mean the system will respond faster with the current error. A too large Kp value may lead to the unstable system, while too small Kp will result in the slow response

The derivative term, Kd, is based on the rate of change of the error so an increase in Kd implies that the rate of change of error has increased over time. Derivative action could not be used alone in practice. This is because its output is only related to the rate of change of the error. The error could be huge, but if it were unchanging, the controller would not give any output. Thus although it is theoretically possible, it is practically impossible.

An increase in the integral term, Ki, means that the error is increased over time. The integral accounts for the sum of error over time. Even a small increase in the error would increase the integral so the robot would have to head in the right direction for an equal amount of time for the integral to balance to zero.c

The following table show the impact of Kp, Kd, Ki parameter on system



**Figure 5: Effects of increasing a parameter independently.**

### Applying PID in our line following robot

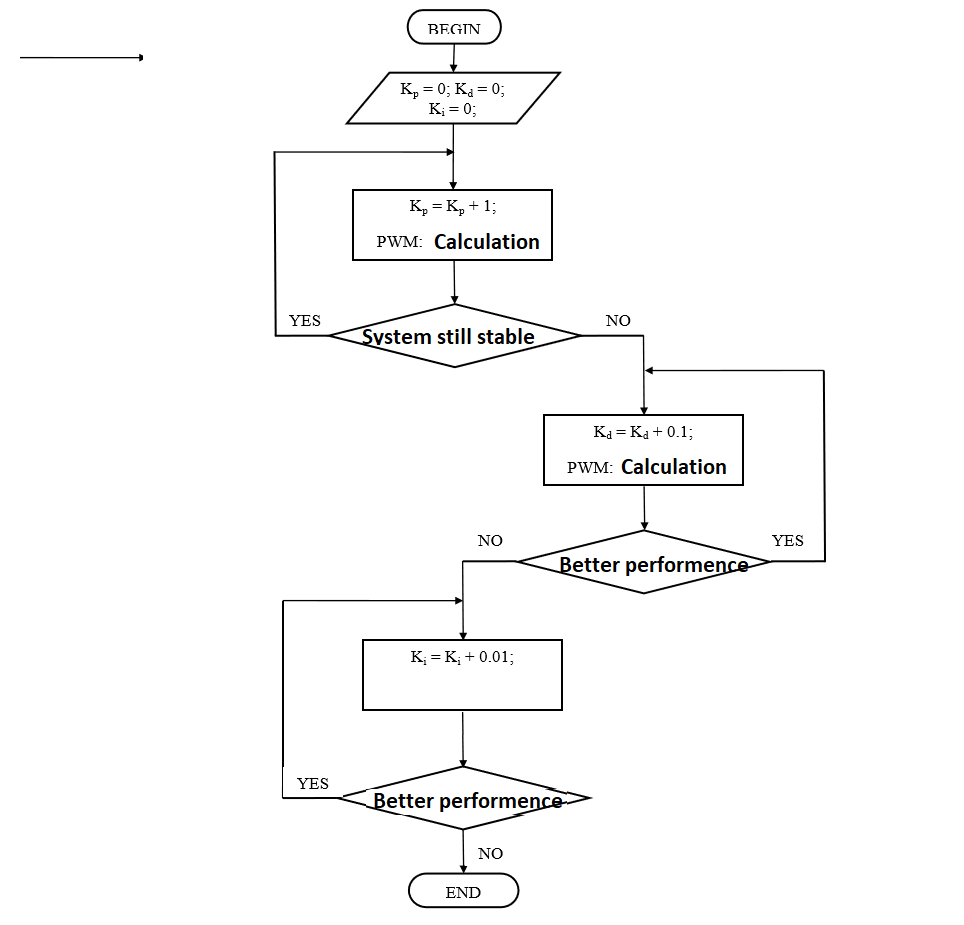
Problem: In line following robot, the steady-state error range is larger than others percise control. It means that if we tuning the Kp and Kd value properly, we can bypass the Ki value and set it equal to 0.

Tuning PID parameter: In this report, we used a well-known tuning method called Ziegler–Nichols.

Firstly, we increase the Kp value until the system become unstable (in our problem, this is when the car become fluctuate and can not follow the black line properly). In real experimentation, we have change the Kp from 1, 2, 3… and realize that when Kp increase above 26, the car can not follow the black line and become unstable. Therefore, we decide to choose the 25 as a optimal value for Ku parameter (Ku is the inital value used to estimate 3 value in next stage .

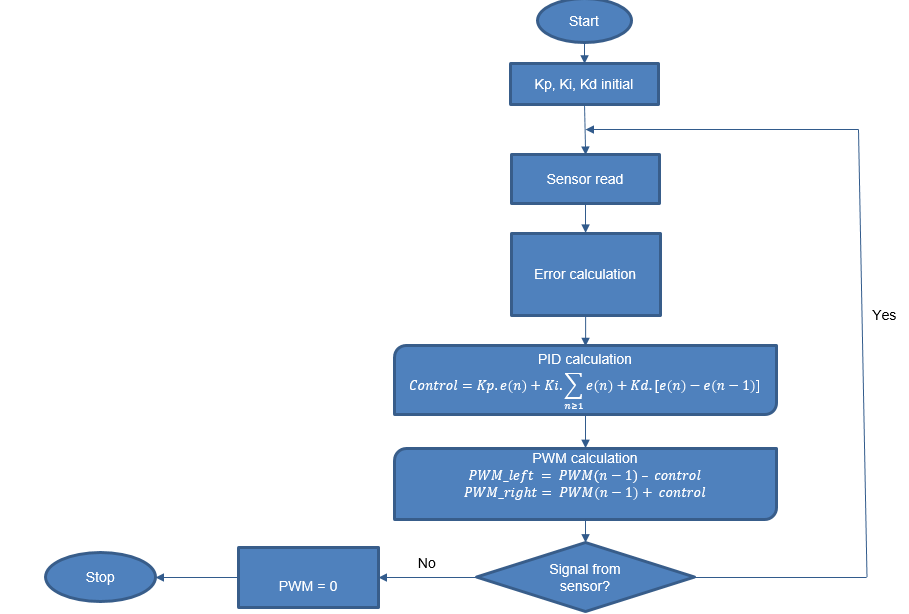
After that, Kp as a base value to calculate Kp, Kd and Ki. Using formulas introduced in [1], we get the proportional, derivative, integral parameter perspectively.

The overall process for PID turning illustrated in the flow chart below



**Figure 6: PID parameter tuning**

After the tuning three parameter, the overall car controller algorithm is illustrated in below flow chart

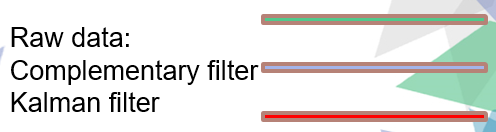


**Figure 7: System controller**

### Different kind of filter applied in our system

In this work, we try to implement different kinds of filter so that we can choose the most suitable method for our application. Firstly, the Kalman filter is used. The sensor is a bit unstable, so the kalman filter is an algorithm which uses a series of measurements observed over time, in this context is our MCU sensor. These measurements will contain noise that will contribute to the error of the measurement. The Kalman filter will then try to estimate the state of the system, based on the current and previous states, that tend to be more precise that than the measurements alone. For the implementation, we use the open source in [2] to applied on our system.

On the other hand, we also apply a complementary filter, which is much more easier for understanding as well as implementation compared to Kalman mentioned above. The complementary filter can be considered as a combination of low-pass filter and high-pass filter. The performace of this implementation is illustrated in figure below



### 2.3 Path finding algorithm

#### 2.3.1 Algorithm description

The main idea of path finding algorithm in our project is reduction of action of robot. In the first time, when the line follower car move on the maze,we use left-hand rule algorithm for finding the way through maze.This method was chosen because it is one of the most straightforward way to implement. In brief, we can be simplified this rule into simple conditions: in a intersection, if you can turn left then turn left, else if you can go straight then go straight, else if you can turn right then turn right, if you are at a dead end then turn around and continue finding the path.The action of robot was implements orderly with priority decrease steadily form turn left to turn right. All action taken at an intersection of the first movement is recorded. In line with recording, the optimization is executed by checking out triad action and reducing if it meet the criterion. The following table shows the mapping of action and character in implement code.

|  |  |
| --- | --- |
| L | left turn |
| R | right turn |
| S | going straight past a turn |
| B | turning around |

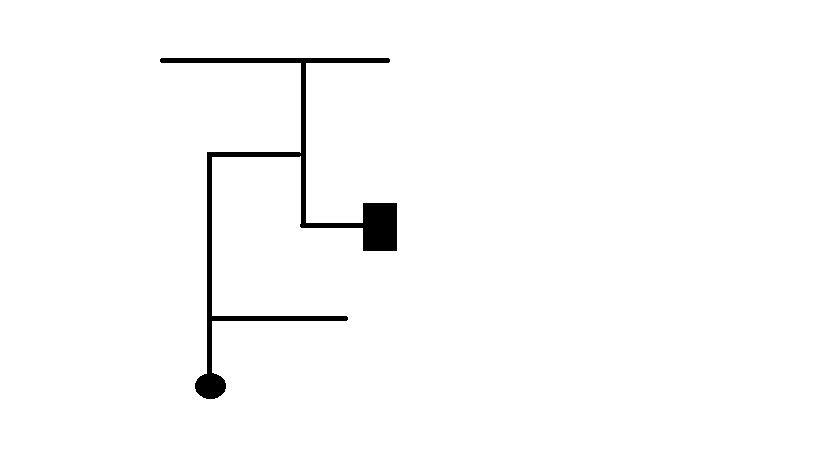
**Table 4: Actions at an intersection**

After set of record has collected,the optimizational path is reached by reduce the subset of abundant action. For instance, when the robot go straight(S) -> turn back(B) -> turn left (R), these state are equal to just one action - turn right (R) . The experiments was then repeated  under conditions in which subset action is mapped with equivalent action. By the way, we get the whole list in table 2.

|  |  |
| --- | --- |
| LBR | B |
| LBS | R |
| RBL | B |
| SBL | R |
| SBS | B |
| LBL | S |

**Table 5: List of triad reduction**

When all subset is reduced, the optimization is done. Here is an example of shorting a specific map.



**Figure 8: Maze example**

The figure 3 indicates a sample maze. Circle symbol stands for Start, and rectangular symbol stands for finish - the way out of maze. Firstly, we get the set of action of robot: SRLLBSBLSL. Then, we reduce the abundant subset action step by step: SRL(LBS)BLSL -> SRLRBLSL -> SRL(RBL)SL -> SRLBSL -> SR(LBS)L -> SRRL. Finally, we got the optimal path SRRL.

#### Path finding algorithm flow chart

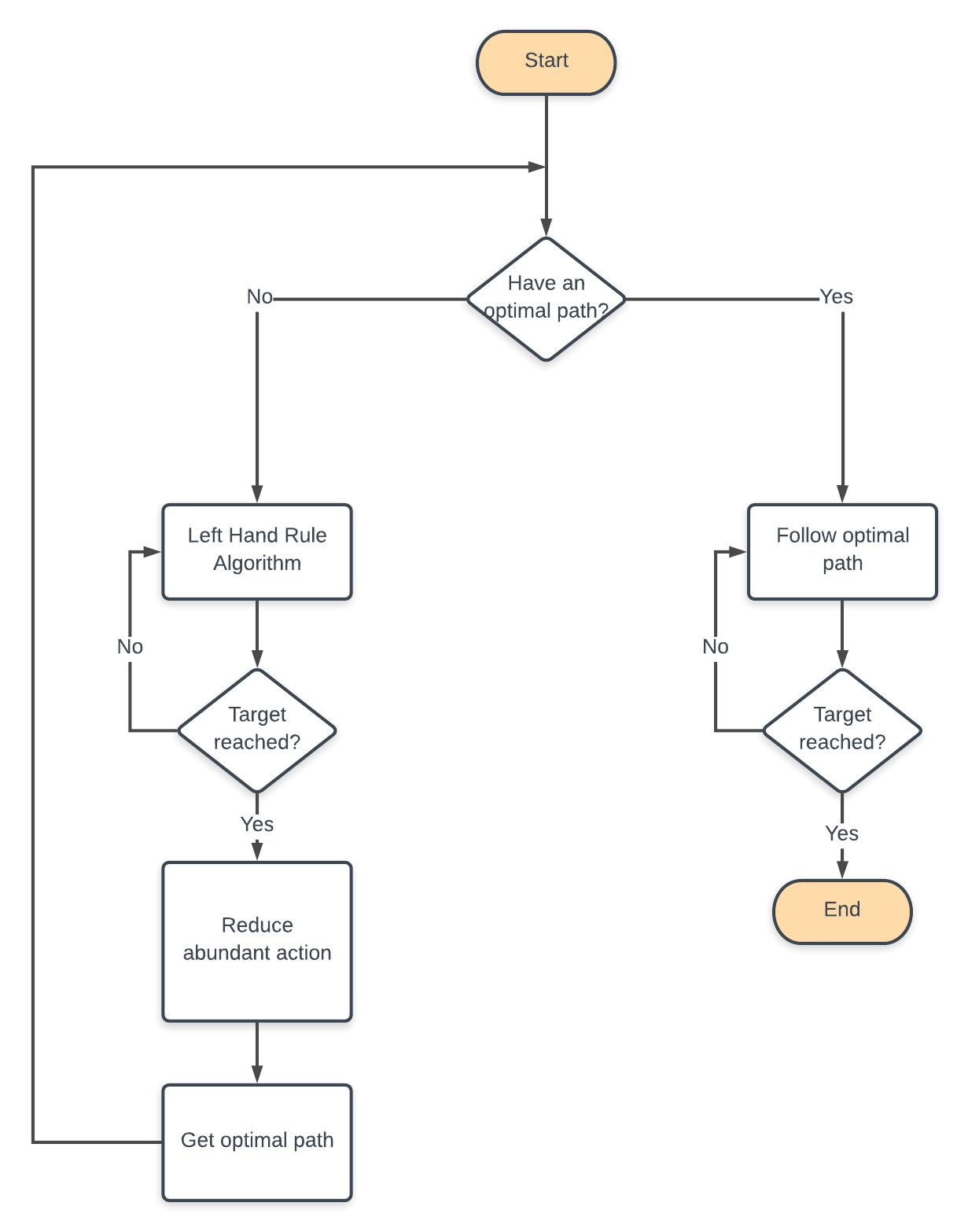


Figure 9. Algorithm Flow Chart

### 2.4 Android application for user control

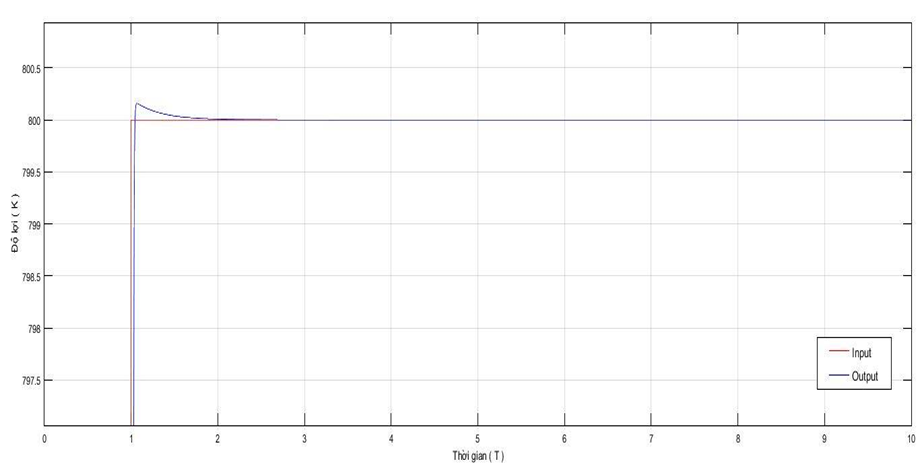
We propose an additional feature for our smart robot, that is the ability to follow the user instruction. This idea can be achieved by an android application, we have used the HC06 module for the communication between uses smart phone and the robot.



**Figure 10. Android app for robot control**

# Results

We have found find an approximation transfer function of system is 2/s(0.3s+1) and simulate it on matlab follow the Nichols-Ziegler process. The result show in figure below.



**Figure 11. Matlab simulation results**

In matlab simulation, we found the PID parameter are 18 - 0.14 - 0.05, which are not suitable for our system (the car can not follow the line). So, we try to find the parameter by hand as demonstrated above (the final result is 15 - 0.001 - 5)

|  |  |
| --- | --- |
| Value of sensor | Error value |
| 11110 | 4 |
| 11100 | 3 |
| 11101 | 2 |
| 11001 | 1 |
| 11011 | 0 |
| 10011 | -1 |
| 10111 | -2 |
| 00111 | -3 |
| 01111 | -4 |
| 11111 | -5 or 5 (depending on the previous value) |

**Table 6: Mapping sensor value with corresponse error**

# IV. References

[1] J. G. Ziegler and N. B. Nichols: Optimum Settings for Automatic Controllers, Trans. ASME, Vol. 64, 1942, s. 759-768

[2] <https://github.com/TKJElectronics/KalmanFilter>

[3]<https://www.academia.edu/6261055/Complementary_Filter_Design_for_Angle_Estimation_using_MEMS_Accelerometer_and_Gyroscope>