3105 Report

Yip Lung Fei 19052736d

This report will briefly describe what I did in the semester 2 of 3105, the difficulties I encountered and how I resolved them.

Demo1

What demo1 does is to allow the car to be remotely controlled and to activate the ultrasonic sensor. For remote control, it is natural to have a remote control, and the design of the hardware is to use 6 ports to control the 8 buttons of the remote control. This requires some skills, I followed the lecture note to complete the programming of the remote control.

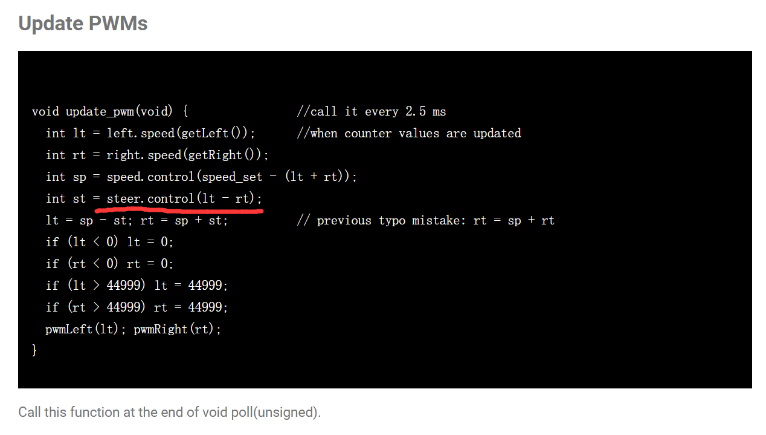
The Bluetooth accessories are already paired before they are handed to me, so I only need to deal with the Usart part. The things received by the Usart on the remote control side will be sent to the car's Bluetooth accessories through the Bluetooth accessories. Specifically, when it is determined that the button is pressed, Usart sends a character, so the programming on the car side should write how to move the car when this character is received.

Demo2

I spent the longest time in demo2.

The requirement of demo2 is to trace. After demo1, I found that it is very unstable to give PWM directly. For example, I want the car to go in a straight line, because each motor is different, even if I give the same PWM, the behavior of the two motors is different, and it can't go in a straight line at all; for example, I found that it needs more torque when starting, When I want the car to go slower, I will give a smaller PWM, but this PWM may not be enough to start the car, and the car can continue to move with this PWM only after I assist the car to start. If I use a larger PWM, although the car can be started, the speed may be faster than I want, which is very difficult to control; for example, when changing the environment, the previous code will fail. My map is an example, I folded the map so it could fit in my backpack for portability, but found that this would create creases and make the map bumpy. The car can walk on the flat map in the lab, but when I switch to my own map, the car gets stuck in the creases. The program written in this way can only be applied to this specific car, this specific map, as long as other conditions change, the program will fail. I don't think this is a good design, so I decided to use PID to control.

However my grasp of PID is very low, I only know the general theory, but not the application. Fortunately, Mr. Lam, the laboratory assistant, has the sample code. He has written all the settings and frameworks of the car. As long as I understand his code, I can use his code directly.

Since I almost don't know anything about the application of PID, I found a lot of videos explaining PID. Through these videos, I know that PID is responsible for controlling speed and differential speed. With these videos and reading Mr. Lam's code, I gradually got some understanding of his code. 'speed' is used to control the speed, the speed is calculated by the wheel counter, and ‘steer’ is responsible for controlling the differential. An error needs to be given in the Control() function. Then I found a problem. The error in the steer is the gap between the left and right wheels. When the gap is 0, the car will go straight. However, in this task, I need to turn, because the track is not straight, which means that the left and right wheels should have errors, so how to set this error has become the first problem that bothers me.

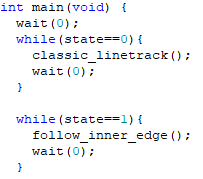
I continued to look for information, and finally found a tracking car made by a student of South China University of Technology on Bilibili. He open sourced his code, so I studied his code and found his way to set the error of the differential pid.



Basically, if the car deviates from the track, give a 1 or -1 error to make the car turn left or right. He uses the exhaustive method to write all the possibilities, so his judgment sentences are very long. I extracted his idea, and then used the control function written by Mr. Lam, combined the two and simplified his idea. My idea is to use the middle two sensors (a total of eight), when the middle two sensors both read 1, it means that the car is on the track, so error=0, if one of the middle two sensors is 1 and the other is 0, it means The car is off track, giving an error=1 or error=-1. When the error is 1, whether to turn left or right depends on how Mr. Lam's code is written. I found through experiments that the car turns left when error=1. When the two sensors in the middle are 0, it means that the track has been deviated a lot. The other 6 sensors are used to determine whether to turn left or right, and at the same time increase the error to make the car return to the right track faster. This error is updated every 2.5ms, and then the error will be handed over to the control() function to update the PWM, which is also updated every 2.5ms. At this point, the car has been able to walk along the track (outer circle).

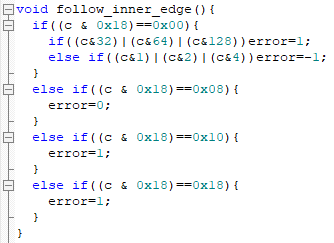
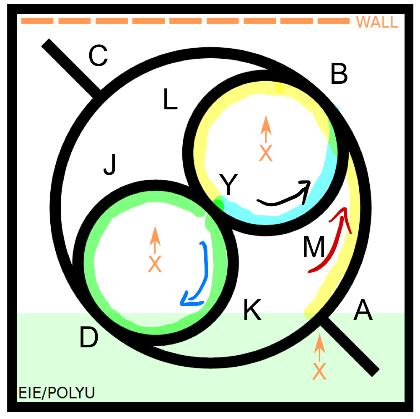
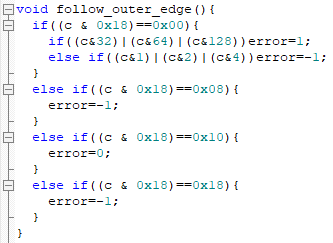
But this task is not just walking along the outer circle, how to enter the inner circle has become my second problem.

I've tried many times, but the car can't get into the inner ring stably, and occasionally it succeeds once or twice, but it's still very unstable. From the current point of view, there were two problems at that time. The first was that the logic was written in a mess, which belonged to the poor programming level. The second is to simplify his code and there will be corresponding problems. The above is what I did in batch 1. Due to my slow progress, I was unable to complete all the demonstrations in batch 1. Lawrence agreed to let me participate in batch 2, during which I gave myself a month off and did service learning, and finally I continued this project in mid-August.

After restarting, I drastically changed the structure of the program. There is a wait(int x) function in Mr. Lin's code. Its function is to execute what I want to do every x 2.5ms. It is responsible for executing background tasks. Updating PWM is one of the tasks executed in this function. In the previous code, there was only a {while(1) wait(1) ;}in my main() function, that is to say, I put everything including determine different states in a single function wait(1), so the logic The judgment is very complicated, and I am very confused when I was writing it. And now my main() is what should be done in different states. It is up to another function to determine which state it is.

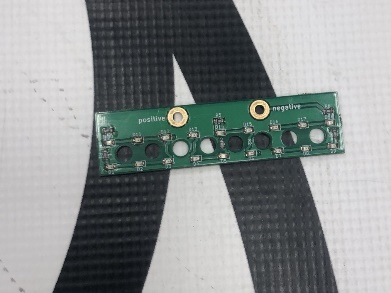
In addition, I also thought of new ways to enter the inner circle. The code of the student I read before should be to select the track by slowing down and giving a larger error. He wrote a lot of conditions, each of which has a corresponding speed and error. And my code is simplified, only error is given, and the speed is constant. My new idea is to trace the edge of the line instead of trace the solid line, there is a difference between these two.

When using two sensors to check the line, it is difficult for me to change the route, because the error should be 0 when the car is on the line, but because it wants to enter the inner circle, it has to give another error, and the program is difficult to write. If I follow the student's idea, I should be able to succeed, but I think it's too troublesome, so I wrote another program called Tracking the inner side, the method is similar to the previous one, the only difference is that when the two sensors are ‘01’, the error is 0 (previously the error is 0 when senser reading is ‘11’), so that it changes from tracking solid line to tracking the edge.

After a circle outside, the car came to point A again, and this time let the car go to the inner line (yellow line). I tried to let the car take the outside line on the road from b to L, but it is easy for the car to not enter the inner circle, so I changed it to the outside line (green line) when I reached point Y. When the car reaches point Y again, take the inner line (blue line), and when the car goes to the fork road that is ready to go to b, let the car follow the solid line, and the task is completed.

I divided this task into different states. When different sensors read a certain value, the state will be changed, and each state car will have different actions. I once had difficulty in changing the state. I went to Lawrence for help. Lawrence said that I used too few sensors. After thinking about it, it was indeed the case. For example, I tried to change the state when the leftmost sensor read 1. But the car didn't go the way I expected it to. This is because I take it for granted that the sensor on the far left of the car will not be used during the tracking process, but it is not. In the end, I honestly used all the sensors, and I was able to successfully change the state according to what I wanted. As for how I know what the sensor is reading at the time, I use this method.



I borrowed the PCB for the sensor from Mr. Lam. The behavior of my car in operation is relatively stable, so I press and stop the car when it reaches the corresponding position, and measure it with PCB. Try several times to find several possibilities, and use them to help determine how to change states.

As for why I use three line-tracking methods, it is because if only one is used, the code will be difficult to write. For example, if I track the solid line, it is difficult for me to change the track to enter the inner circle. If I only use the method of tracking the edge, there will be problems either track the inside edge or the outside edge. If I track the outside, the car will run out at points A and C, and if I track the inside, the car will not be able to complete the first loop running on the outside according to the requirement. In fact, it is possible to write only by tracking the inner and outer edges, but it will be more troublesome in places such as the above, so I finally chose to use three methods for this demonstration.

Demo3

What demo3 does is stop the car before hitting the wall in three different places. Demo3 is the simplest. On the basis of the control system of demo2, I only need to add ultrasonic components and write a condition. There were basically no major difficulties during the period. The only problem was calibration. At the beginning, I measured the distance from the front of the car to the ultrasonic component, and wrote a condition. When the distance is less than a certain length, it means that the car is already away from the wall. Very close, so the motor stalls and PWM is set to 0. I tested it with my hands first, but I found that no matter how I adjusted the distance, the car just wouldn't stop. So, I suspected that there might be a hardware problem, so I opened the test code, and finally found that the distance displayed in the software is not equal to the displayed distance. I was too lazy to change the program, so I put my hand in front of the car to measure the distance displayed in the software, and used this distance as a reference.

When I can stop properly, I start to increase the speed of the car. There are two ideas here. One is to use different speeds at different distances, and the closer to the wall, the slower the speed; the other is to keep the speed at a constant speed. Locally set PWM to 0, after the motor stops, the car still has inertia and will move, and the friction is used to complete the braking. I went with the latter because it only takes a few sentences procedurally to do it. As for that distance needed to stop between car and wall, it was measured with several tests.

Demo4a

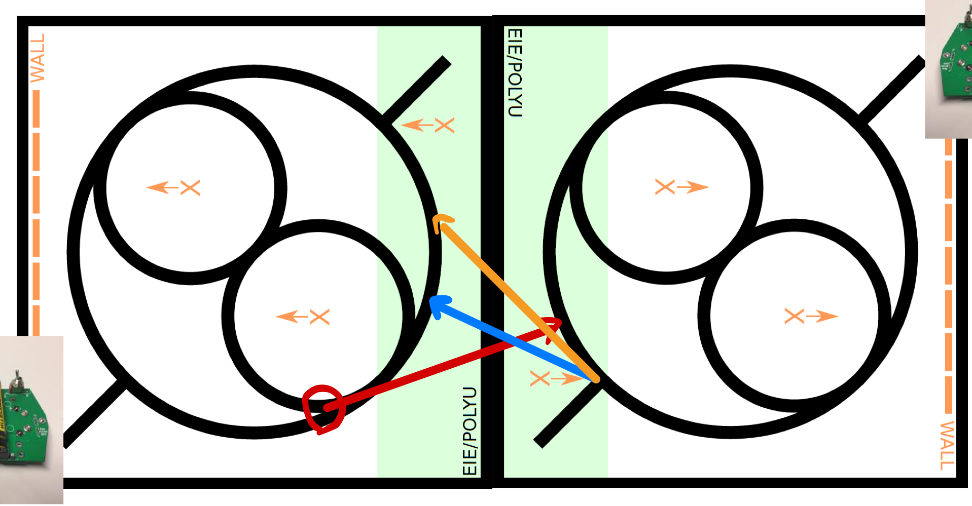
The content of 4A is that two cars pass the ball, and one person takes turns to pass the ball to each other, a total of 6 times. (Three hits per car) This was supposed to be done by two people and before demo2, but since I couldn't find a crew member, I put him last.

4a has no difficulty in programming on the basis of the previous one. I rewrote the demo1 and used the control system to make the control more convenient. The difficulty of 4a is that the road is uneven. I mentioned before that my map has creases, and the ground is not smooth (a friend made it in the parking lot, the concrete is very rough), so the trajectory of the ball is unpredictable. I tried to hit the ball on the opposite side, but because the road was uneven, the ball rolled into the green area in the middle, and the rules stipulated that the car could not enter the green area, resulting in failure of the standard. There is also wind in the parking lot, and the wind can also affect the path of the ball and so on. Even though my car is easy to control, this demonstration is the hardest to do. I have tried many times before I succeed. It can be said that success is due to luck.

Before that, I thought about doing it myself. There are two cars, so two remotes are needed, but I only have one Arduino board (the AVR controller controls the remote by default). So, what should I do? I think of two methods. The first is to replace the Arduino with an ARM controller, but the setting of the arm is too troublesome, so I am too lazy to do it. The second is to manually replace the Bluetooth components, but I will be in a hurry, and the Bluetooth connection will take time, which will make the grades very poor, so I finally decided to ask my friend for help.

Demo4b

4b is a relay race. Before I started, I thought the difficulty was how to get off the track and enter another map. But in fact, this is not difficult. On the basis of demo2, the sensor reads certain conditions and goes straight. After getting started, I found that there are still two places that are more difficult to do. The first is that his next behavior after going to another map is not necessarily the same as I expected.



For example, if I let car1 start to walk straight when it sees the intersection of the red circle, then the path of car1 is shown by the red line. But after going to another map, according to my line track logic, the car may go either counterclockwise or clockwise. Similarly, car2 also wants to go to another map. The condition I gave is that the sensor reads the protruding line and starts to go straight. I expected that he would go on the tangent of the circle, which is the orange line, but in fact the car went It's the blue line. If the car takes the blue line, there is a chance to enter the inner circle. This is that my conditions are not set, so I tried to change the conditions here, but let the car go a short distance along the trajectory and then go straight (using the counter method), so that he will go the orange line. However, it has not been working, and when the car reaches the back, it will not go in a straight line, and it will not be able to pass another picture. I've been stuck here for a long time, I thought it was a problem with my code, but in the end, it was able to pass by trying another car, so I went to Mr. Lam to change the car. Then it worked. It is worth noting that when changing the map, there is a black line on the border of the map. This should also be taken into account when writing the conditions. The second difficulty is the steering of the car. There is a part of the car that needs to be turned about 180°. My code is often out of control at this stage. I think it is PID’s problem. I don’t know what went wrong. It may be the accumulation of errors too big.

When I try it with Mr. Lam's code, it doesn't lose control. (In demo2, I asked Mr. Lam about PID related things, and Mr. Lam gave me the complete sample code he wrote for reference). Mr. Lam use different method to got the error, and the tracing logic was also different with mine, but the frame of my PID came from Mr. Lam. So is there something wrong with my method? Just because my code is cobbled together? I am so confused. In the end, I didn't solve the problem of PID out-of-control, my solution was to use Mr. Lam's code for the turning part. The problem comes again, because the tracing logic of Mr. Lam and mine is different, and I think my tracing method works better (Mr. Lin's code will be inaccurate when the speed is slightly faster), So I didn't want to keep using his code for line-tracking, I just wanted to use his code for steering, and finally I came up with a solution to combine the two systems, which is clumsy but effective.

Conclusion

I have gained a lot in this project, including the knowledge I learned when looking for materials to solve problems, including practicing problem-solving skills because I solved a lot of problems, including knowing that instead of trying to solve everything by myself, it is better to solve problems by myself. when seeking help. I experienced the joy of success and realized my own inadequacies. For example, I know that if I didn't use Mr. Lam's microcontroller settings, my task would be much more difficult. After all he set usart, timer and other settings for me. On the other hand, the students of South China University of Technology, he does all the setup himself. Although I used PID, I also checked the method of tuning PID parameters, but in the end I didn't tune it very well. The most I tune is the p term, and the i term has also been tuned a little, and I did see how different i's affect the movement. As for the d term, I wrote it casually. After demo2 was completed, I tried to increase the speed. I originally used the speed of 5. When I tried to use 7 and 8, the movement oscillated a lot and could not complete the request. This is the consequence of not adjusting the PID. Although I said that I don't want to use PWM to control the car directly, because this is only applicable to this car, but in fact, using PID control also needs to adjust the corresponding parameters according to different models. But since the requirements were met, I was too lazy to continue tuning. And if I had to rewrite the PID control system myself, I probably wouldn't be able to write it, because in fact, I didn't understand many things in Mr. Lam's code, such as the meaning of those variables in his code for calculating speed, etc.

I only know the running process and general principle of PID control now. There are still many places I can do better, such as trying to find why the car is out of control, such as improving the logic of judging and changing state to make the success rate higher, etc. But the time is limited, and my level is also limited. At present, I can meet the requirements and it is enough for this project. At least I know that I am very interested in this aspect. I enjoy the process of solving problems and enjoy the feeling when I solve the problems successfully.