实验报告：激光雷达智能车

完成者姓名： 薛俊智 **学号：** 523030910124

1. 实验目的及应用背景

本实验旨在通过实际搭建和编程控制一个集成了麦克纳姆小车、激光雷达和ESP32控制板的智能车系统，来探索和实践智能车辆的感知与控制技术。通过这一实验，我们希望达成以下几个目标：

1. **理解麦克纳姆小车的工作原理**：通过实际操作麦克纳姆小车，理解其全向移动的机制，并掌握其控制方法。
2. **掌握激光雷达的运用**：学习如何使用激光雷达进行环境感知，以及如何将传感器数据用于避障和路径规划。
3. **编程ESP32控制板**：通过编程ESP32控制板，实现对智能车运动的精确控制，并理解其在智能车辆系统中的核心作用。
4. **综合应用智能车技术**：将上述技术综合应用于原地避障、环形赛道和迷宫赛道三个不同的场景，以验证智能车系统的实用性和可靠性。

应用背景方面，智能车技术在自动驾驶、智能交通系统、服务机器人等领域有着广泛的应用前景。随着技术的发展，智能车系统需要能够更准确地感知环境、做出快速决策并执行精确控制。本实验模拟了智能车在复杂环境中的运行情况，为未来智能车技术的研究和应用提供了基础。

通过本实验，具体来看，我们能够将实际学习的理论知识与系统实践相结合，感受并弥合理论与实践的差距。长远来看，我们不仅能够加深对智能车技术的理解，还能够为未来智能交通系统的发展做出贡献。

2. 实验主要器材

2.1 麦克纳姆小车

麦克纳姆轮小车是一种全向移动平台，它通过四个角上的麦克纳姆轮实现全方位移动，包括前进、后退、侧移、斜移和自转。这种小车在自动化仓库、服务机器人等领域有着广泛的应用。 本次实验通过控制四个轮子以不同的速度方向旋转来实现小车的全方位运动，在实验初始阶段我编写了如下的麦轮运动解算的函数方便我们进行后续的运动控制。

```
void setspeed(float x, float y, float w)//正:  x向前，y向左，w逆时针。
{
  float fl,fr,bl,br;
  fl = x + y - w;
  fr = x - y + w;
  bl = x - y - w;
  br = x + y + w;
  mecanum.driveAllMotor(fl, fr, bl, br);
}
```

2.2 激光雷达

激光雷达（LIDAR）是一种使用激光光束进行距离测量和检测的传感器。激光雷达以一定的角速度匀速转动，每旋转一周，收集到的所有反射点坐标的集合就形成了点云。在智能车项目中，激光雷达用于环境感知，能够提供周围障碍物的精确距离信息，帮助小车实现避障和路径规划。 本次实验采用Rplidar A1雷达，特点如下： • 二维平面12m探测范围 • 360度全方位测距，生成点云地图 • 三角测距系统，低成本 • 适用于室内环境和无日光直接照射的室外环境 • 扫描频率5.5Hz 采样点的数据帧主要信息格式如下：`((float)angle, (float)distance)`距离的单位是mm，数据还包含当前采集点是否为一次新的扫描的布尔值。

2.3 ESP32控制板

ESP32是一款低成本、低功耗的系统级芯片（SoC），集成了Wi-Fi和蓝牙功能。在智能车项目中，ESP32作为控制核心，负责处理传感器数据、执行控制算法，并驱动小车的运动。 本次实验中使用Arduino对ESP32控制板进行编程，控制板主要负责：接受激光雷达数据、算法设计、控制麦克纳姆轮电机运动。 Arduino代码框架以及初始化如下,在这段代码中实现了引脚绑定，雷达数据获取的功能，是后续实验的基础：

```
#include <RPLidar.h>
#include "MecanumDriver.h"

// 创建激光雷达对象
RPLidar lidar;
// 创建麦克纳姆轮驱动器对象
MecanumDriver mecanum(9, 8, 12, 13, 11, 10, 46, 21);
// 用于存储激光雷达数据的数组（0到359度共360个值）
float distances[360] = { 0 };

void setup() {
  Serial.begin(115200);  // 开启用于调试的串口
  lidar.begin(Serial2);  // 将激光雷达连接到串口2（19、20引脚）
  mecanum.begin();  // 启动麦克纳姆轮驱动器
}

void loop() {
  if (IS_OK(lidar.waitPoint())) {                                    // 等到一个新的扫
描点
    float distance = lidar.getCurrentPoint().distance / 1000.0;  // 距离值，单位m
    int angle = lidar.getCurrentPoint().angle;                       // 角度值（整数，
四舍五入）
    bool startBit = lidar.getCurrentPoint().startBit;                // 每进入一次新的
扫描时为true，其它时候为false
    if (angle >= 0 && angle < 360) {                                 // 角度值在[0,
359]范围内
      distances[angle] = distance;                                   // 将距离值存储到
数组
    }

    if (startBit) {              // 每进入一次新的扫描处理并控制一次
      //do sth.(主循环)
  } else {
    // 重新连接激光雷达
    rplidar_response_device_info_t info;
    if (IS_OK(lidar.getDeviceInfo(info, 100))) {
      lidar.startScan();
```

```
        delay(1000);
      }
    }
  }
```

3. 实验内容

3.1 原地避障

实验目标：将小车周围划分为八个方位，通过编程实现麦轮小车躲避向它靠近的物体。

实验思路：利用激光雷达，检测周围的障碍物相对于小车的距离和方向，当有障碍物在附近时，小车沿着反方向运动躲避障碍物；当小车附近没有障碍物时，小车静止。

代码摘要：

```
float distance_min = 10;  // 用于存储最近障碍物距离
int angle_min = 0;        // 用于存储最近障碍物对应角度
// 遍历寻找[0, 359]范围内最近障碍物距离及其对应角度
for (int angle = 0; angle < 360; angle++) {
float distance = distances[angle];
if (distance >= 0.15) {           // 激光雷达的最小量程为0.15m, >=0.15的才是有效数据
    if (distance < distance_min) {  // 如果障碍物距离<最近距离
    distance_min = distance;      // 更新障碍物最近距离
    angle_min = angle;            // 更新最近障碍物对应角度
    }
}
}
if (distance_min > 0.5) {             // 如果最近障碍物距离大于0.5m
mecanum.driveAllMotor(0, 0, 0, 0);  // 小车静止不动
} else {
if (angle_min > 22.5 + 45 * 7 || angle_min < 22.5 +45 * 0)  // 最近障碍物位于后方
    mecanum.driveAllMotor(100, 100, 100, 100);                // 向前
if (angle_min > 22.5 + 45 * 5 || angle_min < 22.5 +45 * 6)  // 最近障碍物位于右方
    mecanum.driveAllMotor(-100, 100, 100, -100);              // 向左
if (angle_min > 22.5 + 45 * 2 || angle_min < 22.5 +45 * 3)  // 最近障碍物位于左前方
    mecanum.driveAllMotor(0, -100, -100, 00);                 // 向右后
    ......//这里其余方向逻辑类似不再赘述
}
```

3.2 环形赛道

实验目标：小车通过雷达感知环形赛道，并沿着赛道以尽快的速度最少的碰撞跑完赛道。

实验思路：获取小车左前方和右前方的平均雷达距离，理想情况下，沿着中线行走的小车左前方和右前方的平均距离是相等的，用实际差值通过PID控制小车的角速度达到目标。

代码摘要：

```
float midline(float data[])//中线理论值获取
{
  float res = 0;
  for(int i = 225;i < 255; i++) res += (data[i] - data[360 - i]);
  return res/30.00;
}

float mid = 0.0;float last_mid = 0.0;
float k = 40.20; //80 90 40
float p = -35.00; //50 50 40  参数记录
float x_speed = 27.50;float w = 0.0;

void loop() {
    ......
    float distance = lidar.getCurrentPoint().distance;  // 距离值，单位mm 这里我们为了保留最大精度采取原始数据记录单位。
    ......
    if(startBit)
    {
      last_mid = mid;
      mid = midline(distances);
      float dmid = last_mid - mid;
      w = (-k*mid - p*dmid) / 1000.00; //PID参数控制输出量
    }
    setspeed(x_speed, 0, w);
    ......
}
```

3.3 迷宫赛道

实验目标：将小车放置于泡沫砖块垒出的迷宫里，小车自行判断障碍，最终走出迷宫。

实验思路：让小车始终沿着右侧墙运行，通过“右手扶墙”来解决岔路口问题，走出迷宫。

代码摘要：

计算前方距墙距离平均值。

```
float getfront(float data[])
{
  float res = 0;
  int num = 0;
  for(int i = 170; i < 190; i++)
  {
    if(data[i] != 0)
    {
      res += data[i];
```

```
      num ++;
    }
  }
  return res/(num + 0.01);
}
```

计算右前方和右后方距墙距离平均值。

```
float getrightfront(float data[])
{
  float res = 0;
  int num = 0;
  for(int i = 225; i < 265; i++)
  {
    if(data[i] != 0)
    {
      res += data[i];
      num ++;
    }
  }

  return res/(num + 0.01);//防止/除以0产生nan的错误
}

float getrightback(float data[])
{
  ``float res = 0;
 int num = 0;
  for(int i =275; i < 315; i++)
  {
    if(data[i] != 0)
    {
      res += data[i];
      num ++;
    }
  }
  return res/(num + 0.01);

}
```

主循环实现：我们通过对右前方和右后方的平均距离差值进行PID控制来确保小车直行。与此同时，通过特定条件下对角速度进行比例放大或缩小实现离墙过近进行左转、右转掉头等运行逻辑。

```
float difference = 0.0;float last_difference = 0.0;
float k = 80.20; //80 90 40
float p = -16.00; //50 50 40
float w = 0.0;  float threshold = 200.00;

void loop() {
```

```
  float x_speed = 50.00;
  float y_speed = 0.00;
   ......
    if(startBit)
    {
      last_difference = difference;
      float rf = getrightfront(distances);
      float rb = getrightback(distances);
      float d_difference = difference - last_difference;
      difference = rf - rb;
      float mid = (rb + rf)/2;//右侧距离平均值
      w = -(k*difference + p*d_difference) / 200.00;

      if (getfront(distances) < threshold)
      {
        x_speed = -35;
        w = w > 0 ? 30.00 * w : w / 8.00;//如果前方距墙过近则触发转弯，尽可能向左转。
该条件优先级最高
      }else if (mid < 250.00)//如果右侧距墙过近则向左平行移动。
      {
        y_speed = -55.00;
      }else if (difference > 800.00)//如果右前方距离较大（右侧墙消失），此时应扶着右侧
墙掉头。
      {
        w /= 8.00;//减小角速度、增大转弯半径，防止撞墙。
      }if(getfront(distances) >= 800.00)//前方距离很远，加速直行。
      {
        x_speed += 60;
      }

      setspeed(x_speed, y_speed, w);
    }
  } ......
}
```

4. 实验结果及分析

4.1 原地避障

实验结果及分析：将代码烧写至控制板，能够较好地躲避来自四面八方的障碍并合理做出反应，符合理论预期。

4.2 环形赛道

实验结果：最初由于雷达精度不高、四舍五入等一些问题小车运行会出现超调或者不足的情况。最终在参数调整以及代码优化后能够较好地绕着赛道运行，达到实验目标。

分析：但是受限于雷达频率以及精度，不能十分丝滑地快速无碰撞地绕行环形赛道。这也许可以通过后续代码优化进一步提升效果。

4.3 迷宫

实验结果：最终实现了小车沿着右侧墙运行走出迷宫的目标，但是过程中会产生碰撞以及不能及时转弯的问题。

分析：由于雷达扫描需要时间较长，以及15cm内的距离值无法获取的原因，仅仅通过右前方和右后方的距离差值控制并不能完美地完成任务，后续可以通过改进算法，增加更多逻辑判断来改善小车表现。

5. 实验中遇到的问题及其解决方法

问题一

问题详述：环境配置时，ESP32包下载困难。

解决方法：上网查找ESP32库本地下载。

问题二

问题详述：电脑烧写程序后，右后方麦轮只前转不倒转。

解决方法：更换小车实验发现问题仍然存在，求助老师后猜测问题在于电脑端，发现ESP32库配置出现问题，重新配置后解决。

问题三

问题详述：在环形赛道实验中，小车速度与反应和理论预期相差过大。

解决方法：仔细阅读代码后发现忽略了startBit在数据获取时的作用。完善整体逻辑后解决。

问题四

问题详述：在迷宫实验中，获取并输出右前方和右后方距离平均值发现在正中间时候差异过大，且输出值不稳定。

解决方法：仔细阅读代码对角度的处理逻辑，发现将角度四舍五入之后存在一些点未赋值、一些点重复赋值的情况，增添逻辑判断，仅获取非零平均值后解决。

问题五

问题详述：在迷宫实验中，如果距离墙过近，部分变量输出nan错误。

解决方法：考虑到代码中num可能为0，出现除以0的错误，在num后加上0.01的小量避免报错，并增加0值处理逻辑。

问题六

问题详述：在迷宫实验中，如果需要执行扶墙从右侧掉头，可能会识别为前方遇到墙，表现出左右旋转卡在原地不移动。

解决方法：减小距墙距离阈值改善误判情况，同时将左右转的角速度按比例放大或缩小产生差值解决卡在原地的问题。

6. 实验总结&致谢

本次实验中，我们实际搭建和编程控制一个集成了麦克纳姆小车、激光雷达和ESP32控制板的智能车系统，完成了三项任务。在实验过程中遇到了大大小小的问题，我和队友通力配合，积极寻找解决问题的办法，最终较好地完成了整个实验。我认为本课程以及实验巩固提高了我在智能车感知与控制领域的实践能力与知识水平。深深感谢老师以及助教，还有与我一同协作的队友，希望将来能够在智能车、无人驾驶等领域贡献出我们的力量。