Imagen que contiene Logotipo

Descripción generada automáticamente![Logotipo, nombre de la empresa

Descripción generada automáticamente](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4REwRXhpZgAATU0AKgAAAAgABAE7AAIAAAAjAAAISodpAAQAAAABAAAIbpydAAEAAABCAAAQ5uocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEdlcm3DoW4gQW5kcsOpcyBEaSBGb256byBDYXR1cmVnbGkAAAAFkAMAAgAAABQAABC8kAQAAgAAABQAABDQkpEAAgAAAAM0OAAAkpIAAgAAAAM0OAAA6hwABwAACAwAAAiwAAAAABzqAAAACAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMjAyMDoxMDoyNiAxNjoxMzoyMQAyMDIwOjEwOjI2IDE2OjEzOjIxAAAARwBlAHIAbQDhAG4AIABBAG4AZAByAOkAcwAgAEQAaQAgAEYAbwBuAHoAbwAgAEMAYQB0AHUAcgBlAGcAbABpAAAA/+ELNWh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMjAtMTAtMjZUMTY6MTM6MjEuNDc1PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkdlcm3DoW4gQW5kcsOpcyBEaSBGb256byBDYXR1cmVnbGk8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgAqQDaAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kcUYoooAMUYoooAMUhFLSMQBknA9TQBk6Kwe61bplb0j/xxK18Vx/g3UrW71rxAkE7O32zeAzHBXaBkfiD+ldhW1eDhOz8vyOXCVFUpKS8/zDFGKKKxOoMUYoooAMUYoooAMUYoooAMUYoooAMUYoooAMUYoooAMUYoooAMUYoooAMUYoooAKKKKACiiigBk0yW8Ek0pwkalmOOwrwrxb4yvtd1SZILmSOwVysUSEqGX1b1Jr3cjKkHkV4vrHjbULPW721gtNOEcM7oubUE4BIr2cpjepJqN36nzufStSinPlT7K9/xRx0NxLbyiS3leJ16NGxUj8RXrvw58YyavC2manM0t5HzG5HLp7n1FcP/AMJ/qv8Az6ab/wCAgrvvhxrU2u295Nd29rHJC4VWghCcEZr08yUnQbqU1p1v/wAA8bJ3GOKjGlUevS2j/E7eiiivkz7sKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiori4itbd57iRY4o1LMzHAAFeR+LviRdalI9pocj21p0MynDyf4CuvC4SpiZWh95wY3MKODjeo9ei6npV94l0yxu47Np/Ou5TtS3gG9yfcDp9TiuP8VHwVo10zXmmi6v5m3yRRucgnkljnArBtpP8AhCvC63rANreqLmIvy0EXr9T/AJ6VxMssk0rSzO0kjnLMxySfUmvbwmXLn5oyfL91/wDgHzWPzaTgoyguZ66q/L9/U7NNe8FGRQ/hmVVJ5IlzgfTNd9pV54b0PQzqWjxlbCZg0skILbPdh1GK8MrZ8NeIp/D9+HXMlrJ8txAT8si9+PWuvFZf7SHuN+l3r95wYLNXSqXnFeqS0+7oe9WOpWepW4msLmO4jP8AFG2atCvD9ein8Ka1DqXhu7kisb5fOgKH5R6oR0Irv/Bnjy38QqLO92waiB93Pyy+6/4V4FfAShT9rT1j+K9T6nC5rCpV9hWXLP8AB+h2NFFFeae0FFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUHgUVzfjrXToXhiaSJgtxP+6h9iep/AZNaU6cqk1CO7Ma1WNGnKpLZHAfEfxe+qXzaTYSEWcDYlZT/rXH9BXOeFNL/tjxNZWjD93v3y+yLyf5Y/Gsnr165rr/AAPizsdd1X+K2syiH0Zv/wBVfaOksLhuSnvt83ofnMa0sbjFUq7b/Ja2MjxXq51rxJc3I/1KN5UI7BF4H59fxrGoJycnrSV204KnBQXQ8+rUlUm5y3YUtJRVmZ2Okf8AE8+H+padKd02mn7Vbk9Qv8Q+nWuSgnltp0mt3aOSNgyupwQRXSfD25EPi2K3f/VXkTwOD3yP/rVz99bGy1C4tW4MMrRn8DiuOklGtOm9nr9+531m5UKdXqrr7tV+DPcPBHikeJdH3TYW8gwk6jv6MPY101fPvhDXT4f8RwXRYiBj5cw9VP8Ah1/CvoFWDqGUgqRkEd6+WzLCrD1vd+F7H2+T454uhafxR0f+YtFFFeYe0FFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABXjvxX1Q3PiKGwRv3dpFlh/tt/9bFexV88+Lbs33i7Upycgzsq/ReB/Kvayanz13J9EfOcQ1nDCqC+0/yMiuusP9E+FOpS8g3d6kWfYYP9DXIV199+6+E+mJ/z2vXc++Nwr6PFa8ke8l/mfIYPT2ku0X/l+pyNJS0ldhwhRRRSA0vDs5tvEunSg423KZ+mcVc8bQfZ/GmpLjG6Xf8AmAf61jW0nk3UUgONjhvyNdL8R02+NJnHSWGN/wDx3H9K5JaYmPmn+aO+GuDl5SX4pnK1794G1L+1PB9lMTl0Xyn+q8V4DXq/wfui2m6hak/6uVXA+ox/SuDOKfNh+bsz0uH6vJi+T+Zf8E9Hooor5E+/CiiigAooooAKKKKACiiigAooooAKKKKACiiigAPQ4r5wurDUJLyZ2srolpGJPktzk/SvbvGeuX/h/Rhf6fDFKFkCyCXPAPcY968//wCFt6v/AM+Nn+Tf4172VqvTjKdOKafnbY+WzuWFqTjSrTcWtdr7nF/2bff8+N1/35b/AArrddtLn/hXvh+BLaZmDSMyiNiRz344qf8A4W3q/wDz42f5N/jW7rHj/UNP8OaRqEVtbtLfKxdWBwuPTmvQrVMU5wvTW+mvkzyKFHBKFTlqvbX3fNeZ5f8A2bff8+Vz/wB+W/wpP7Nvv+fK5/78t/hXqFp4p8aX2kHUrXRbV7fG5eoZh6gZ5rFi+K2uTTpDFp9q8rttVArZJ9OtaxxWJlflgtN/eMZ4LCQs5VJK+3u7/icT/Zt9/wA+Vz/35b/Cj+zb7/nyuf8Avy3+Fen6v4t8Y6HYpd6hpFmkLYBZSTtJ6A88VR0n4heJtbvhaabplpLLjceoCj1JzxSjjMRKPOoxt/iHLA4WM1Tc5cz6cv8AwTz7+zb/ALWVznt+5b/Cur+INjdXGt2c8VrPJ5lhFuKxMcHng4HWtjWPiB4n0K7+zanplpE5GVIyQw9Qc81d8TfEDUNFOnC3trd/tVos7b88E9hWLr4mdWE1Bdba7nRHD4OFGpB1Ho1f3dt/M8w/s2//AOfG6/78N/hXoXwlgubbUtRW4t5oleJMGSMrnBPr9aof8Lb1f/nys/yb/GrFj8UNbvtQgtYrKz3zSKi8N3P1q8T9brUZQcEk/MnB/UcPiI1Y1G2unL/wT1mikXO0buuOcUtfIH34UUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUARXMMdxbvFMiyIykMrDINfNMo2zOBwNxxX004ypx6V4DL4N8RGZyNIuSCxwdor6DJqkIc6m7bfqfKcRUZ1PZuEW99l6GDXX+Jv8AkR/DP/XOT+YrL/4QzxF/0CLn/vkV0/iDw5rFz4Q8P28GnzPNbo4lQLymSMZr2K1ak6lO0lv38mfP4fDV1Sqpwey6Puj0zSUWPQrJUACiBMAf7orxzw/Gg+Kka7RtF7JgY4GN1djbeJvFlvZxQDwnIwjQJu3nnAxXJWOk+JrLxQuspoM7sJml8o9Oc8Z/GvIwlN01V5pL3lpqj3sfWVZ0OSL91q/uvy8j1Dxuqv4K1QMAcQEjPqK4z4PKPM1RyBuxGAe+PmqzrGu+K9X0e5sG8LSRi4jKFwxO38KyPCSeJ/ChufK8OzXH2jbndxtxn/GopUXHBzpNq7a6ryLr14zzCnXUXypa+6/PyND4wIPL0x8fNucZ9sCue8ffe0P/ALBkdafiweJvFcdssvh2a3+zliNpznP/AOql8aeGtYvW0j7Jp803k2CRybQDtYdRXZhZRpRpRnJXXN1Rw46Mq8q86cXZ8vRnn1d98JLeKXXrySSNWeOAFCRnbk84rnP+EN8Rf9Ai5/75Fdv8MdC1PSdVvZNSspbZHhCqZBjJzXVj61N4WajJX9ThyvD1VjabnB2v2fY9LFFA6UV8WfowUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFRXMJuLeSJZZIS64EkZAZfcZzzXgWh+J/FV/8AFFfDdz4jvPsq3skLMoQMyoW/2epxXTRw7rRk09tTnrV1ScU1ufQVBry/4vXut+HtLs9S0HV72GSe4W3aBdrKfkJyBjOflqD4TfEiXxJbyaHr90f7TUFoZzhTMvp/vD+VX9Um6Pto6ol4mKq+yejPVsilrzjQf7Yb4r6ppNzr99cWFhbxzRxSFMuX7MQvIHtiuJ8Y+JvE2ifE1tBsvEV6LR5oVUsELKJMZ/h7Zp08I6k+RS1tcU8UoR5mutj32ivLvGemeNPDGhy6zoniu6vVtRvmguIYydncjjt6elQ+FviBdeOfB2pwS3T6brWnwGbzbXAEgCnDYIPGRyKn6rJw9pFpof1mKnySVmer0YFeHfCLxJ4i8V+KJYtY127kgtYfOEShAHO4DDcdOa6rxxJrNr468PWuna5eWlrq0zRTRJsIXaAflyvGfxpzwkoVfZN67hDEqdP2iWh6PRXimq+PbzWfiSPDaa5Joek20jQSXK7RJK69SWI4yRit/VfC3iy21TTV8P8Ai7U5rC8lK3EkoSQwLtLBgcDIOMfXFOWEcLc8rXVxRxKlfkjex6ZRXh/xMvvFPgRNONp4svrs3ZkDeZGi7duPQe9bFpY+MbvwPp3iCw8UX91czLFM9n5KEMpYbgCBnpR9UtBT5lZ7biWJvJw5Xdeh6xmivn/4ieK/E/hjxw+mad4gvBbMkcgVwhK7uoztrY+IV54u8BQ2Go2Xim4vYLh9hjuIY+GxnsORVrAyfL7y97bcn65Fc10/d3PaKK47wvrtz44+HSag0kunXTqytLbEAh1OMrkHg46V5v8AC7xV4l8U+NFsdW167e3hhaYogQbypAweOnNZwwk5Rm27cu5pLExTiv5tj3misLxlry+HPCd7qHWZU8uBe7SNwo/MivI/hn441+L4iHRvE+oT3CzB4DHO2fLlByMfkR+NTSws6tOVRdB1MTCnNQfU95orlviJJeWngu+1HTNQuLG5s4jKhhIw/swIPFcL8Hde13xXqV9LrWtXU6WQQpD8oVy2fvcZ7UQw7nRdW+iHKuo1VTtqz2OisvxLcy2nhfUZ7aQxyx27MjjqpA61jaXoN9faTa3UviLUleaJXIDJgEjP92ohSThzydlsKpXlGp7OMbu1zraK5LXp73wx4d22+oS3FxdXCQpcXWD5W7jPAA4p7+E737L5lv4i1H7btysjyAoT7rjGKr2Mbczlo9jN4mfNyRhdrV6r+mdVRWZ4fub660WF9WgMF4MrKpGMkHGR7HrWnWEouMnFnXCSnFSXUK+bvC//ACcI3/YTuf8A2evo24WV7d1t3WOUj5HZdwU+pGRmvMNP+Dlzp3ipfEEPiNmvVnaclrQbWZid2Ru75Nd+Dq06caim7XVjjxVOdScHFbO5tfEjBm8KBun9uwZz9GrzD4o+CrnwZ4ij8SaDujs5ZxIpQf8AHvLnOPoe35V614x8G3viubTymsfYUsJluI1S33EyjoxJPT2rdudKTVNCfTdc8u7WaMpMQm1W9wMnH506OJVBQs773RNXDus5XVtrM8y+FfiQ+K/H2r6s8XlTSadAsqjpvBIJHtxmuP8AiT/yXaP/AK72n8lr1HwF8M18Dapf3UOom6W6jEao0W3YASRznnrWXrvweuPEHiaTXLrxC0d2zqy+XagBNuNuPm7YFdFPEUIYiU07RtZGM6FaVCMWtb3Ox8dXcFl4D1ia5cIn2R1yT1JGAPxJrxL4NaXdXE2v30UbGGPTZINwHDO3IX/x39RXpWr/AAz1LxMYovEvi27u7SMg/Z4YEiVj6nHeuw0Pw/pvhzSk0/SLZYLdeoHJY9yT3JrnhiKdChKnF3bNpUZ1aynJWSPEP2fyF8XagpIDGz4B6n5xXovjU+b8Q/BMCcyC7mk2/wCyI+TWZffB02/iZta8Ja5Lo0zsW2CPeEJ6heRwfQ10Xh3wTLputNrev6tNrOq7DHHLIoVIUPUIo6ZrXEV6U6vt0+m3ysRRpVIU/ZNdd/mch40+Etr4uu59d8LahCs87t5sbcxSODhiCOhyDn3rmfhLr+uaL4+XwtfSSvbu0kclu7bvJdVJyvoOMfjXpX/CG+ItN1C/k8NeJUtLS9nac2s9oJBEzHJ2nPHJqbwf8ObPwxqFxqtzdSajq1ySZLqVQMZ67R2zQsVFUJU6kuZW07oTw8vbKcFbXXscN+0P/q9C+s3/ALJXo3w7/wCScaF/15p/Ksfx58NpvHV9BJc6z9mt7YERQpb5wTjJJ3c9K6LwnoVz4b8PwaVcXwvUtl2QyeVsITsDyc1jVq03hIU09UbU6c44mU2tGeE/Gj/kqh/64Q0vxH1XVP8AhOrXT/GRNxpdq4eJLdfL8yJurD/awMfhXoXif4QTeKvEUmr3+vbJmChUjtQFVV6D71bfi74eR+MtDtLbU7xUv7U5W8igAyO4256HjvXbDGUYqmm9lZ+XoccsLVk5tdXf1N7R4tMh8K266Csa6f8AZ8wCPptIz+deEfAr/kok3/XlJ/6EtereF/A2q+FdBuNKsvECywSA+UZrXJhJ6kfN09qyfCPwjm8H6+mqWGvea+wxuklsMOpxkfe46da56dWlTp1Yc1+bbc6KlOpOdOXLtuN+IN9c6x440bQNM0+TUhp5Go3dukioHwcIpJ49T+IrzH4jf2npfj6LXpNKk0ee4ZbiNGlWTLpjJyvHUD869u0LwZe6T4w1DX7jV1u5NQG2WM2+3aB90Kd3AHH5VX8ffDxvHT2gm1JbSK1yUCwbmJPXnPTgcYqsPiaVKpGL+G2r167kV8PUqxcut9Ng8W6pHrXwav8AUocbLnT/ADBjtnGRXDfs9f8AHxrn+7F/7NXZW/w71G28ByeFU19TaSEjzGtcsqE5Kj5vX+dUPDfwp1PwnJcPofip4DcACTdZq+cdOp96mNSjHD1KSlu9NypQqutCo47LU6/xbcQt4X1e3WVDMlozNGD8yqc4JHocH8qoaJoN5LoNjIniDUIlaBCEXy8Lx0GVpNO8FXMFjrKaprMmo3erKEe5eILsUAgAAdhk1Yt9A120to7eDxFtjiUIo+xocAVhGcY0+SMlv1Xl6Cq05Sre0lBtWto7dfVFi9tNPi02HSdfupL1b2Ty42uByzdQMqAB7VS/4RK/0+P/AIkOv3cG0fLDcYlj+nPIFXJ/Dk2paTJaa3qDXUm8SQzxxiNoWHQjHeq40fxOsPkDxDEUxgSm1HmY+ucZojOysprzutPloTUp3d5U3to09V6u5Z8Ka1cavZXCahGkd5ZztBN5f3WI7j863qztE0aDQ7D7Nbs0hZjJLK5y0jnqxrRrlquDm3DY78PGpGlFVHqMllWGFpH3FVGTtUsfyHJrDj8baHKGMU8zhDtbbayHafQ/LW/XAeFb+5sv7eW30ya8A1KY7o2UAHA45IrWjSjOEm91brY58VXnSqQino79G9vQ6aTxXpMQiJnkbzohKnlwO+VPQ8Dj8arp440GSNpI7mVkQ4ZltpCF+p21p2MMX2OO4W3WGSSFdygYxxnH6mua+H0ST6FqMcqhke+mVge4JqlTpOEpWeluvf5ESq1/aQgmveTe3a3mb934i0+zlWJ3lllZN/lwQtIwX1IUcfjSS+JdLh0ldTedvsbEjzBEx24ODkYyORjmubvjq/hTxFe6lbaedQ0y7CGXyz88O0Y/KpdZv9P1P4Z6jd6Sf3EqMxGMFWLZYEduTVrDwbi1qm1r6/kZvGTSmnpKKbs122a7o3m8TaYmkjU2klFmT/rfIfGPXp09+lXNP1G31WzW6si7Qt91mQru9xkdK5i//wCSQt/2DV/9BFbfhb/kU9L/AOvWP/0EVlUpRjTcl3aN6Vacqqg9uVP5kj6mV8Rx6Z5Yw9sZzJn0YDGPxqbUtUtdIszdX7MkKn5nVGYL9cDisuX/AJKFB/2Dn/8AQxUnjP8A5E3U/wDrgaSpxc4R72G6s40qk+sb2+SL9jqtrqVh9ssy7wYyG8phuHsCMn8Kr2XiTTdQW4Nm80n2bPm4t3G0+nTr7daj8Jf8ifpX/Xqn8qreFgPO13Ax/wATOT/0FaHTgufy/wAxxq1Gqb/mWv3XHx+NdCljaRbqTy0ba8hgcKp9CcYFWr/xLpWm2cV3dXOLaUZSZEZ0P4gECuP8KXsVvoWswNby3Ms19cBIY4y2/oMZ6D8TUkOhzab8PbTTtVUFp7yPfFnOwNIPl/KumWGpRnZ33t6o4YY2vKnzKz0b9Hfb5nd2t3Be2kdzauJIZVDIw7g1QTxLpcusPpcVwZLxDh40jZtv1IGBXMeF9VbQdJ1jSr4kyaQ7eSD1ZG+4B+P8xVfwrYyab8RLmCc7pm09ZJm/vOxBY/man6rFc93stPP+rmn16cvZKK1k7Py/ppnolFJRXnnri0UmaM0ALRSZozQAtFJmjNAC0UmaXNABRRRQAyVnWJjGm9wPlXOMn61yOgaf4g0U6hvsLWb7XdPcAi527d3Y/Lz0rsaK1hVcIuNtznq0FUnGd2mtvmZ1mdSTS3e9jSS8YswiR8KATwoOOwxzWV4O0rUdFt7q31CCJVlnedZI5d33j90jH6101IaPavlcbbh7Bc8Ztu8f1MN5tftZ7iGOwivYncmGY3ATap7MCM8e1Zc/ha9t/BFzpFmIprm8dnlbfsRGY5446DGK7GirjXlG1kuj+4ieFhO/M29GvRPc5a50vVJvAf8AY62sIuWtxbk+d8oAGN2cfpWt4ftrqy0K0tL2NI5beJYzsfcGwMZrToNTKs5R5WutyoYeMJqabva3yOfl/wCShwf9g5//AEMVr6hZR6jp89nP/q50KN9CKyJP+ShQf9g9v/QxXQCnUbTi12QqMVJTT2bZyGj2/ibQdPTSlsre/hh+WC68/ZhewZSM8e1bWladLpOkyJhbi7kdppSDtDyMcn6DtWpRROs59Frv5hTw0aaSTbtor9P68zmPB+lanoq3kF/bxBLm5e4Ekcu7bux8uMVc8TWl/fWtvFp0EcjJOkzGSTaPlYHHQ9a26WiVaTqe0a1COGjGj7FN2/E5XUPC0uoeKrDVuIIwgF5CHzvKnKD35/kKIdJ1VPH02stbxC1kgFvtE3zAA/exj9K6qin9Yna3lb5E/U6XNzLvzfMKMUUVznYGKMUUUAGKMUUUAGKMUUUAGKKKKACiiigD/9k=)

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES

MÁSTER EN AUTOMÁTICA Y ROBÓTICA

GUIADO Y NAVEGACIÓN DE ROBOTS

TRABAJO FIN DE ASIGNATURA

Autores:

Álvaro Benito Oliva (M20159)

Germán Andrés Di Fonzo Caturegli (M20037)

Juan José Jurado Camino (M20039)

Tutores:

Fernando Matía

Miguel Hernando

Paloma de la Puente

Madrid, 17 de enero de 2021

Tabla de contenido

[1 Objetivo del trabajo y reparto de roles 3](#_Toc61316871)

[2 Planteamiento del problema 3](#_Toc61316872)

[3 Sistema de locomoción 4](#_Toc61316873)

[4 Sistema de percepción 6](#_Toc61316874)

[5 Estimación del estado 8](#_Toc61316875)

[6 Control 11](#_Toc61316876)

[6.1 Algoritmos de seguimiento de trayectorias 11](#_Toc61316877)

[6.2 Implementación del algoritmo pure pursuit 12](#_Toc61316878)

[6.3 Implementación del control reactivo 13](#_Toc61316879)

[7 Planificación de trayectorias 14](#_Toc61316880)

[7.1 Implementación del algoritmo RRT 14](#_Toc61316881)

[7.2 Resultados de la planificación 15](#_Toc61316882)

[8 Demostrador final 16](#_Toc61316883)

[9 Conclusiones y observaciones 16](#_Toc61316884)

# Objetivo del trabajo y reparto de roles

El principal objetivo que se busca con este trabajo es familiarizarse con los temas más importantes que surgen en el momento de programar la navegación de un robot en presencia de incertidumbre:

* Sistema de locomoción del robot móvil (odometría).
* Elección de los sensores exteroceptivos.
* Algoritmos para la estimación del estado del robot (localización).
* Algoritmos de control de movimientos.
* Algoritmos de planificación de trayectorias.

Para ello, se decide escoger una aplicación de navegación de un robot móvil y se implementará en simulación mediante MATLAB y Apolo.

El reparto de roles utilizado para la consecución de este trabajo ha sido el siguiente:

* Germán Di Fonzo: mapa del entorno, calibración y localización.
* Juan José Jurado: control y localización.
* Álvaro Benito: planificación de trayectorias y localización.

# Planteamiento del problema

La aplicación que se ha decidido escoger para este trabajo consiste en un robot encargado de limpiar el suelo de un restaurante o cafetería con dos comedores. Este robot podría resultar muy útil cuando los clientes ya se han ido y el restaurante está a punto de cerrar. De esta manera, mientras los camareros recogen y limpian las mesas, el robot podría ir limpiando el suelo.

Para poder simular en Apolo el movimiento del robot a lo largo del comedor del restaurante, se ha programado un mapa del entorno con el lenguaje XML en un archivo denominado *entornoRestaurante.xml*. Este mapa es el que se muestra en la *Figura 1* y se caracteriza por ser una sala cuadrada de 16x16 metros con 7 mesas y un muro en medio que separa dos comedores. La mesa de mayor longitud se corresponde con la mesa donde se atiende a los clientes, mientras que las otras mesas se utilizan para que los clientes puedan sentarse y comer.

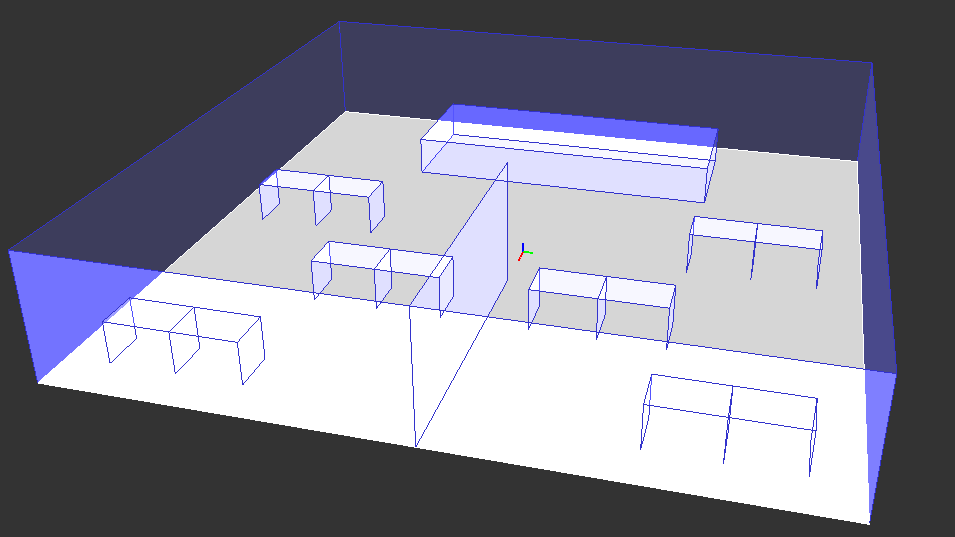


Figura . Mapa del entorno visualizado en Apolo.

# Sistema de locomoción

El robot con el se trabaja en este proyecto es un Pioneer3AT. Este robot se caracterizar por tener un sistema de locomoción tipo diferencial.

En el simulador facilitado ya viene contenido el modelo cinemático del sistema de locomoción de tipo diferencial. Por lo tanto, para la calibración de este sistema se ha seguido el siguiente procedimiento:

1. Se coloca el robot móvil en el origen de coordenadas del mundo, el cual coincide con el centro de la sala cuadrada. La orientación inicial del robot es de 0 rad.
2. Se reinicia la odometría a un valor de [0 0 0] con la función *apoloResetOdometry()*. De esta manera, al colocar el robot en el origen de coordenadas del mundo, el vector *[x y theta]* proporcionado por la función *apoloGetOdometry()* coincidirá con la estimación de la pose del robot según la odometría.
3. Se definen los parámetros del movimiento del robot de manera que describa una elipse. Estos parámetros se han elegido de manera que sean muy parecidos a los valores con los que se trabajará en los algoritmos de localización y control del robot.

* Velocidad lineal: 0.5 m/s.
* Velocidad angular: 0.3 rad/s
* Tiempo del movimiento: 0.1 s
* Número de iteraciones: 1000

1. Se calcula la pose real del robot con la función *apoloGetLocationMRobot()* y la pose estimada según la odometría con *apoloGetOdometry()* en cada iteración. Estos valores de pose real y estimada se van almacenando para calcular posteriormente el error absoluto cometido en cada variable de estado. En la *Figura 2* se puede observar la trayectoria que describe el robot según la odometría (elipse de color azul) y la trayectoria real (elipse de color roja). Como se puede observar, la incertidumbre de la odometría es bastante notoria.



Figura . Representación de la posición real y de la posición estimada con la odometría en cada iteración.

1. A medida que se mueve el robot también se va calculando para cada iteración el avance y giro del robot con respecto a la iteración anterior, tanto los valores reales como los obtenidos según la odometría. Estos valores también se van almacenando para después poder conocer la incertidumbre cometida por la odometría en la estimación del avance y el giro del robot.
2. Se calcula el error absoluto cometido para cada variable de estado del vector de estado, es decir, el error cometido en *x*, el error cometido en *y* y el error cometido en *theta*. En la gráfica de la izquierda de la *Figura 3* se muestra la evolución de los errores en *x* e mientras el robot describe la trayectoria elíptica. Por otro lado, la gráfica de la derecha de la derecha de la *Figura 3* se puede observar el error cometido en la estimación de la orientación.

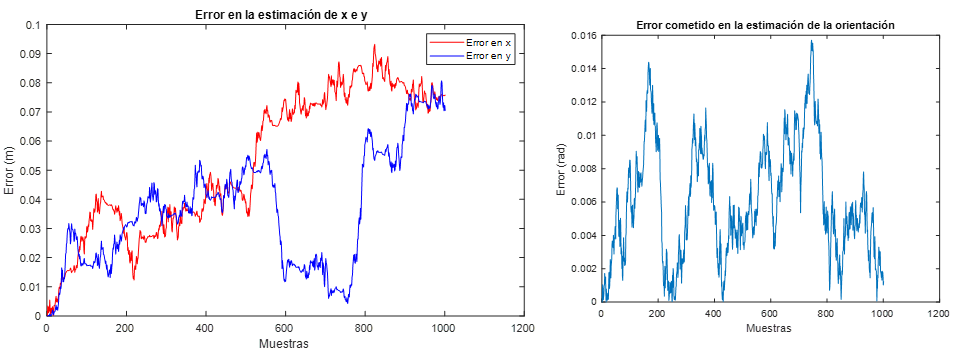


Figura . Error absoluto cometido en la estimación de [x y theta] para cada una de las 1000 muestras.

1. Se calcula el error cometido por la odometría al estimar el avance y el giro del robot durante su movimiento. En la gráfica de la izquierda de la *Figura* 5 se muestra el error absoluto del avance, mientras que en la de la izquierda se observar el error absoluto cometido en el giro.

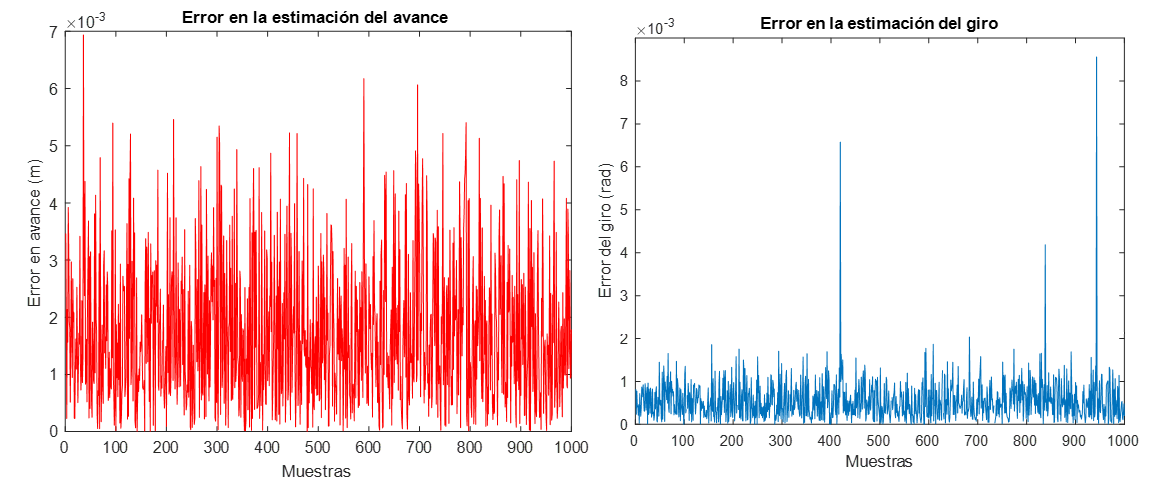


Figura . Errores cometidos en la estimación del avance y del giro según la odometría.

1. Finalmente se calculan las medias y las varianzas de cada uno de los errores calculados hasta ahora. Para ello, se ha programado un pequeño código en MATLAB que muestra por consola estos valores. En la *Figura 8* se muestra un ejemplo de los resultados que se obtuvieron con los parámetros de movimiento definidos en el punto 3 de este apartado.

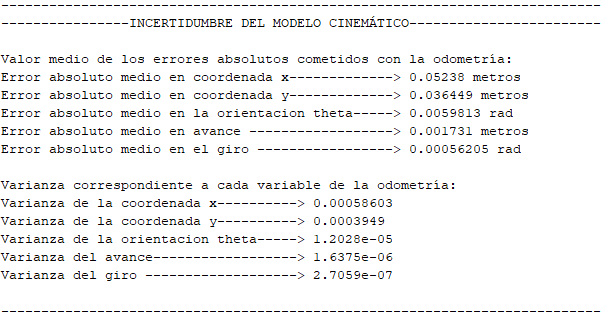


Figura . Medias y varianzas de los errores cometidos con la odometría.

Los valores de las varianzas de los errores absolutos cometidos en la estimación de las variables *x*, *y* y *theta* se utilizarán para la inicialización de la matriz P del filtro extendido de Kalman implementado en el apartado 5 de este documento. Esta matriz se corresponde con la matriz de varianzas y covarianzas de la estimación del estado (matriz de incertidumbres) e indica si la estimación es buena o no.

Por otro lado, los valores de las varianzas de los errores absolutos cometidos por la odometría en el avance y el giro del robot se emplearán para la inicialización de la matriz Q del filtro extendido de Kalman, ya que se corresponde con la varianza del ruido del proceso.

# Sistema de percepción

En este apartado se especifican los sensores exteroceptivos que se utilizan para la localización y el control reactivo del robot. Para el caso de la localización se ha optado por utilizar un telémetro láser LMS100 y 10 balizas colocadas de forma concreta en las paredes del entorno. Por otro lado, para el caso del control reactivo se ha decidido utilizar 3 ultrasonidos, de esta manera se podrán detectar obstáculos que se encuentran cerca del robot y se podrán esquivar.

Para realizar la calibración y así poder estimar la incertidumbre de estos sensores se ha seguido el siguiente procedimiento:

1. Se coloca el robot cerca de la posición [-2, 0] con una orientación de π rad. De esta manera, el robot se encuentra cerca de la mesa donde se atiende a los clientes del restaurante y mirando hacia ella. Esto permite calibrar los ultrasonidos.
2. Se colocan 2 balizas de tal manera que sean visibles por el telémetro láser del robot. En la *Figura 6* se puede observar la colocación del robot en el mapa del entorno y la localización de estas dos balizas.

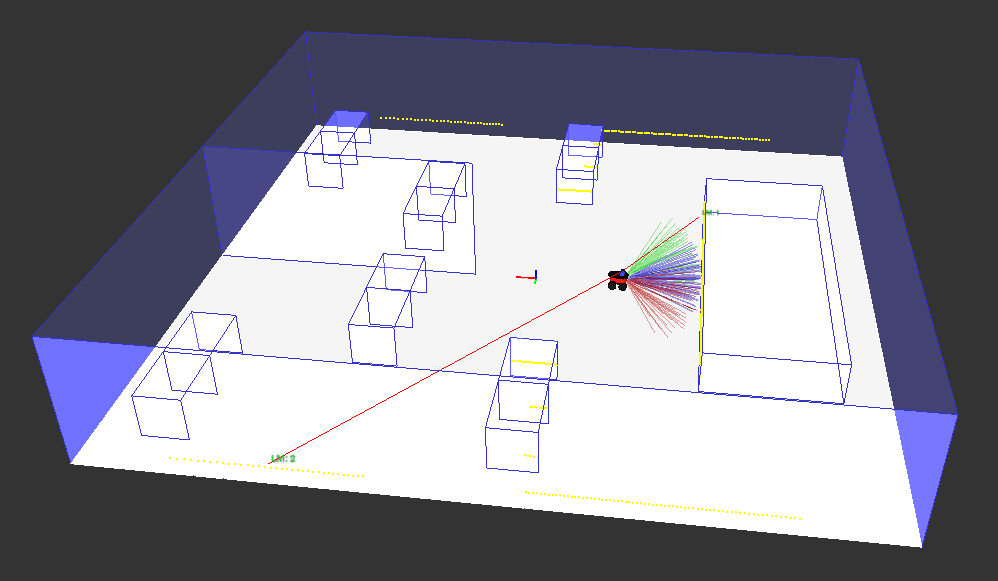


Figura . Posición del robot para la calibración de los sensores.

1. Se mide 3000 veces con cada uno de los tres ultrasonidos la distancia a la que se encuentra la mesa respecto del robot. Además, también se mide 3000 veces la distancia y el ángulo a la que se encuentra cada una de las dos balizas respecto del robot.
2. Se calcula la media y la varianza de las 3000 medidas realizadas por cada sensor. Los resultados obtenidos se muestran en la *Figura 7*. Como se puede observar en dicha figura, la varianza de los ultrasonidos es insignificante, del orden de 10-30. En cuanto a la varianza del ángulo del telémetro láser, esta ha sido de gran importancia para definir la matriz R de varianzas y covarianzas de la predicción de la medida del láser.

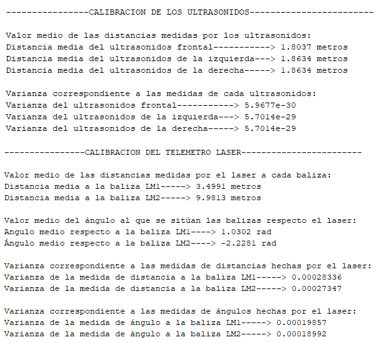


Figura . Ejemplos de valores obtenidos en la calibración de los ultrasonidos y el telémetro láser.

# Estimación del estado

Para poder implementar algoritmos de control y planificación de trayectorias es necesario estimar el estado del robot a medida que se van moviendo por el entorno. En este trabajo, para realizar esta estimación, se emplea el filtro extendido de Kalman.

A continuación se explica el procedimiento que se ha seguido para la implementación en MATLAB del algoritmo de localización que utiliza este filtro.

1. Inicialización de los parámetros del filtro.

* Se coloca el robot con una pose inicial determinada y se resetea la odometría para esa pose.
* Se inicializa la estimación del estado con la función *apoloGetOdometry().*
* Se inicializa la matriz de varianzas y covarianzas de la estimación del estado. Para ello, se utilizan los valores de las varianzas de los errores de estimación de , de y de , obtenidos en la calibración del sistema de locomoción (apartado 3 de este documento).
* Se inicializa la matriz *R* de varianzas y covarianzas de la predicción de la medida a partir de las varianzas obtenidas en la calibración del sensor láser (apartado 4 de este documento). Cabe destacar que el modelo de observación utilizado hace uso de dos balizas, utilizando para la predicción de la medida los ángulos y una de las distancias que estas proporcionan. De esta manera, conociendo la posición de cada una de las balizas se pueden calcular las medidas que el telémetro láser debería aportar.
* Se inicializa la matriz correspondiente a la varianza en la medida. Para ello, se han utilizado los valores de las varianzas de los errores de avance y giro obtenidos en la calibración de la odometría (apartado 2 de este documento).
* Se almacena el valor de las 10 balizas que se han colocado en el mapa del entorno. Estas balizas se han situado en zonas concretas del mapa de manera que el robot pueda ver al menos 2 de ellas la mayor parte del tiempo que se encuentra en movimiento.

1. Movimiento del robot y detección de balizas.

Después de la inicialización de los parámetros del filtro, se ejecuta el algoritmo de localización en bucle. Dentro del bucle, lo primero que se hace es mover el robot con la función *apoloMoveMRobot()* y se comprueba el número de balizas que el láser detecta con la función *apoloGetLaserLandMarks()*. Si se detectan al menos 2 balizas, entonces se ejecuta el código correspondiente al filtro extendido de Kalman. En caso contrario, el robot se sigue moviendo y únicamente predice la posición por medio de la odometría; es decir, sin la corrección por medio del filtro.

1. Predicción del estado

Para la predicción del estado () se utiliza el modelo cinemático del sistema de locomoción de tipo diferencial:

Esto significa que la predicción del estado depende de la estimación anterior y del avance y giro que ha realizado el robot según lo que mide la odometría. Para calcular el avance y giro respecto la pose anterior se utiliza la función *apoloGetOdometry()*.

Por otra parte, para conocer la matriz de varianzas y covarianzas de la predicción del estado se utiliza se aplica la siguiente ecuación:

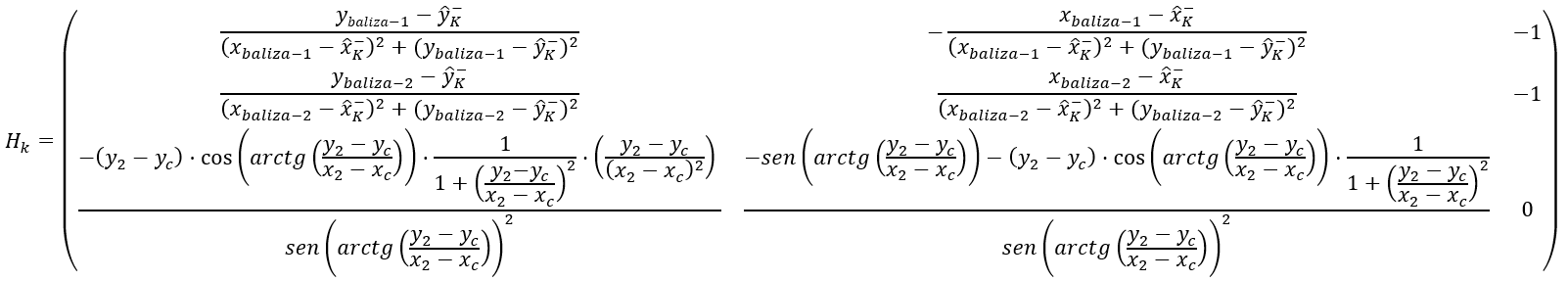
Donde y se corresponden con matrices jacobianas de la matriz , debido a que se trata de un modelo no lineal y hay que linealizarlo. La matriz jacobiana se obtiene al derivar las ecuaciones del modelo cinemático con respecto a las variables de estado (, y de ), mientras que la matriz jacobiana se halla derivando las ecuaciones del modelo cinemático con respecto a las variables *avance* y *giro*. El resultado obtenido de realizar estas derivadas es el que se muestra a continuación.

1. Predicción de la medida del sensor láser.

Como se ha comentado anteriormente, para la localización del robot el láser lo que hace es medir los tres ángulos a los que se sitúan las tres balizas respecto del robot. Para la predicción de la medida del telémetro láser en función de la predicción del estado, se ha utilizado el siguiente modelo matemático:

Como se puede observar, la predicción de la medida, además de depender de la predicción del estado, también depende de las coordenadas e de las balizas que se están detectando por el láser. Por este motivo, en el algoritmo de Matlab, en la inicialización de las posiciones de las balizas se ha utilizado una matriz de 10x2, donde cada columna representa la coordenada e *,* y cada fila la posición de una baliza, de tal manera que se han colocado en orden creciente del parámetro “id”. Gracias a esto, como la función *apoloGetLaserLandMarks()* también obtiene el “id” de las balizas detectadas, se puede conocer la posición de cada una de estas.

Para calcular la matriz de observación que relaciona el estado con la medida es necesario calcular el jacobiano de la matriz con respecto a la predicción del estado, es decir, con respecto a las variables , y . Si se realizan estas derivadas parciales se obtiene la siguiente matriz jacobiana de observación:



1. Observación de la medida del sensor láser y comparación.

En esta parte del algoritmo se almacenan los valores de los ángulos y la distancia a los que se sitúan las balizas respecto del robot, según el telémetro láser (), para poder comparar estos valores con los que se han predicho en el paso anterior con la matriz . Si tras esta comparación el error obtenido es superior a un umbral, entonces se descarta la medida y no se utiliza para la corrección de la estimación del estado. En caso contrario, se calcula la ganancia del estimador WK y la matriz SK de la siguiente manera:

1. Corrección de la medida de la predicción del estado.

En caso de que el error obtenido tras la comparación realizada en el paso anterior esté dentro del umbral establecido entonces se procede a corregir la estimación del estado y su matriz de varianzas y covarianzas mediante las siguientes ecuaciones:

Para comprobar si el algoritmo funciona correctamente, se ha programado que el robot móvil describa una determinada trayectoria en el simulador Apolo. Se ha ido almacenando en cada iteración del movimiento la estimación del estado obtenida con el filtro extendido de Kalman y la pose real del robot obtenida con la función *apoloGetLocation()*. De esta manera, se ha podido representar en una gráfica de MATLAB los diferentes puntos de la trayectoria que ha seguido según la estimación del filtro y según Apolo para poder establecer una comparación visual. Por último, también se han representado las varianzas acumuladas de la estimación del estado.

Los resultados obtenidos se presentan las *Figuras 8* y *9*. Como se puede observar en esas gráficas, los resultados obtenidos de la estimación son bastante acertados.



Figura . Comparaciones gráficas entre la estimación del estado y la pose real.



Figura . Varianza de x, y y theta durante las 500 iteraciones del bucle.

# Control

## Algoritmos de seguimiento de trayectorias

Para llevar a cabo el movimiento del robot a lo largo de una trayectoria generada por un planificador, es necesario programar un controlador que vaya siguiendo los distintos puntos de la trayectoria. El sistema de control utilizado se denomina “*Pure Pursuit Algorithm*” y está implementado el Matlab, lo cual facilita su implementación en el trabajo.

Sin entrar en mucho detalle, el algoritmo mencionado consta de las siguientes etapas:

1. En primer lugar, se obtiene la pose del vehículo a través del algoritmo de localización que se ha programado previamente.
2. A continuación, se busca el punto en la trayectoria que se desea seguir, que se encuentra a una distancia determinada “*look-ahead distance*”. La determinación de este punto se hace a partir de cálculos geométricos.
3. Posteriormente, se calcula la velocidad de giro necesaria para llegar al punto deseado, finalizando la iteración.

Dado que se debe especificar una velocidad de avance constante, el algoritmo de control trabaja únicamente con la velocidad de giro. De esta manera, existen dos parámetros que se pueden modificar para mejorar el comportamiento del controlador: la distancia “*look-ahead*” y la velocidad de avance.

## Implementación del algoritmo pure pursuit

La programación de este algoritmo es sencilla ya que existe una clase en Matlab denominada “*controllerPurePursuit*” que contiene los métodos para la estimación de la velocidad angular que debe tener el robot.

Los parámetros que hay que definir para inicializar el controlador son: la trayectoria, la velocidad angular máxima, la velocidad de avance del robot y la distancia “*look-ahead*”. Además, hay que especificar el error máximo permitido, ya que el algoritmo programado no llega al punto deseado, sino que se queda en una zona del entorno.

Se ha generado una trayectoria sinusoidal que debe seguir el robot y se ha comprobado el funcionamiento del algoritmo variando los parámetros de velocidad lineal y distancia *look-ahead*, obteniéndose las trayectorias mostradas en la *Figura 10*.

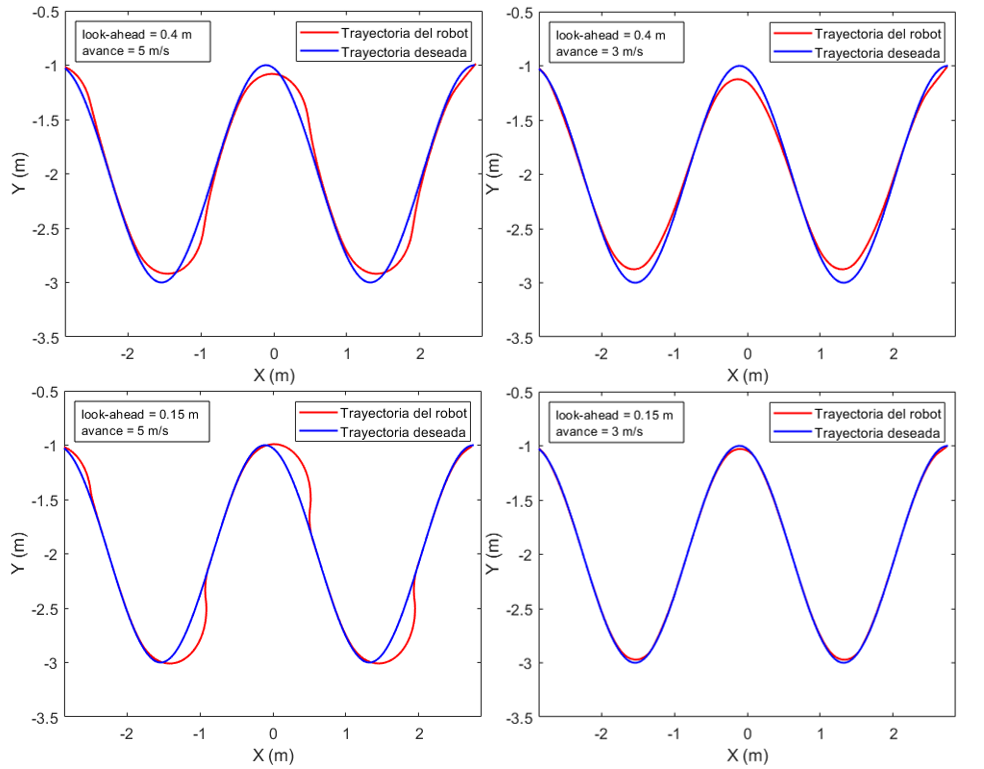


Figura . Trayectorias simuladas para distintos parámetros del controlador.

Como se puede observar, la elección de los valores del controlador es esencial para su correcto funcionamiento. Por otro lado, si se escoge una velocidad de avance elevada, el control se dificulta a la hora de realizar giros bruscos, por lo que es conveniente mantener una velocidad de avance limitada.

## Implementación del control reactivo

La función del control reactivo consiste en evitar colisiones con obstáculos que se presenten en el camino. Es un método de control que otorga cierta autonomía al robot sin un conocimiento a priori del entorno donde se encuentre. El principal problema de este método de control es que se pueden alcanzar mínimos locales de los cuales no se pueda salir fácilmente.

Por lo tanto, la arquitectura de control reactiva entra en ejecución cuando se detecta un obstáculo cercano. Estos obstáculos no se han tenido en cuenta en la planificación porque no se encuentra esa información en el mapa del entorno. Una causa son los obstáculos móviles, como el paso de personas.

Como es lógico, es necesario tener en cuenta estos obstáculos para mantener un control seguro del robot. Este tipo de control se encuentra, por tanto, en un nivel de autoridad superior que el control jerárquico y actuará únicamente cuando se detecten obstáculos que superen cierta distancia de seguridad.

Los sensores utilizados en este trabajo para que gobiernen el control reactivo son sensores de ultrasonidos, debido a que se encuentran disponibles en Apolo su implementación es muy sencilla. Además, en el proceso de la calibración se observa que la varianza de los ultrasonidos es del orden de los por lo que se puede asegurar (en este caso de uso) que su incertidumbre asociada es 0.

Se ha optado por usar tres sensores de ultrasonidos en la parte delantera del robot, de forma que uno gobierne los obstáculos que se puedan encontrar de frente, otro los obstáculos a la derecha y otro a la izquierda (ver *Figura 11*). Así se consigue una mayor precisión del entorno y por tanto un control más sencillo. Debido a que el uso de varios ultrasonidos puede producir interferencias entre ellos, se han separado una cierta distancia para solventar este problema.

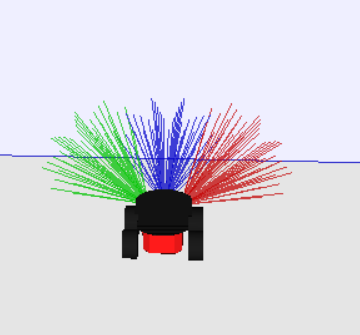


Figura . Uso de tres ultrasonidos para detección de obstáculos.

El código se ha realizado en MATLAB y contempla una condición de precaución, cuando se detecte algún obstáculo a menos de 0.8 metros, y se impone una velocidad de giro y avance determinadas. En caso de que esa distancia de seguridad se vea reducida a 0.3 metros o menos, se detiene el avance del robot imponiendo únicamente una velocidad de giro. Debido a la naturaleza de este control, para escoger los valores de velocidades impuestas, se ha llevado a cabo un proceso iterativo.

La principal característica de este código es que se imponen unas velocidades en función de las distancias medidas por los ultrasonidos. La velocidad de avance será proporcional con la distancia frontal, de modo que a mayor distancia a los obstáculos mayor rapidez en el avance. El razonamiento para la velocidad de giro es el opuesto, inversamente proporcional con las distancias medidas a los lados. Así se consigue evitar el obstáculo más rápido cuanto más pequeña sea la distancia medida.

Para comprobar el código elaborado con los parámetros finalmente escogidos, se ha ejecutado varias veces sobre el entorno diseñado añadiendo diferentes obstáculos estáticos y móviles (otro Pioneer3AT también con control reactivo). Finalmente, los resultados han sido satisfactorios consiguiendo un control autónomo sin que su movimiento se encuentre impedido por obstáculos en su trayectoria.

# Planificación de trayectorias

Para poder llegar desde un punto inicial hasta un punto final sin chocar con ningún obstáculo en el camino, es necesario generar una serie de puntos en el entorno que determinen la trayectoria que debe seguir el robot para lograr su objetivo.

Existen múltiples algoritmos para obtener los puntos deseados, entre los cuales destacan los planificadores discretos, los planificadores basados en combinatoria y los planificadores basados en muestreo. Entre todas las opciones, se ha escogido un algoritmo RRT que construye árboles de exploración para encontrar la trayectoria deseada.

El método RRT está basado en muestreo y está incluido en Matlab, facilitando su implementación para el trabajo. Además, presenta la gran ventaja de trabajar en el espacio de acciones en vez de en el espacio de estados, manteniendo las restricciones cinemático-dinámicas (kinodynamics) del robot no holonómico con el que se trabaja. Esto último asegura que los puntos generados por la trayectoria se pueden alcanzar por el robot móvil.

Sin embargo, este algoritmo presenta el problema de que no se tiene la orientación inicial del robot, pudiendo dar lugar a un inicio de la trayectoria no realizable desde la pose inicial. Este problema se puede solucionar desde la etapa de control, pero sería deseable que el planificador tuviera en cuenta esta situación y pudiera planificar una trayectoria posible para cualquier orientación inicial.

## Implementación del algoritmo RRT

Para llevar a cabo la planificación, en primer lugar se debe importar el mapa del entorno de trabajo, que se ha obtenido a partir de la planta del restaurante. A continuación, dado que el robot tiene una anchura determinada y el planificador trabaja con puntos discretos, es necesario ensanchar los obstáculos para evitar que el robot colisione con ellos. Este ensanchamiento toma como valor la mitad del ancho del robot. Después de esto, se genera un mapa de ocupación de celdillas y se escala para coincidir con las dimensiones del entorno.

Una vez establecido el mapa, se genera un espacio de estados [x, y, θ] utilizando la función de Matlab “*stateSpaceSE2*”. Dicho espacio de estados contiene la física involucrada en el movimiento del robot basándose en distancias euclídeas e interpolaciones lineales para calcular traslaciones y rotaciones.

Por otro lado, hay que establecer un elemento denominado “*validador*” que se encarga de detectar las posibles colisiones que se produzcan durante la generación de la trayectoria. Este objeto comprueba la validez de cada estado generado por el planificador con una resolución determinada por el usuario.

Tras esto, se crea el planificador a partir del espacio de estados y del validador. Además, se debe establecer la distancia máxima que tendrán las aristas de los árboles generados y el error permitido en el punto final planificado respecto al punto final deseado.

Finalmente, se ejecuta el planificador y se obtiene la trayectoria de puntos que llevan al robot móvil desde la posición inicial hasta la posición final.

## Resultados de la planificación

Tras un proceso iterativo de ajuste, se ha obtenido los parámetros mostrados en la *Tabla 1.*

|  |  |
| --- | --- |
| Dimensiones del mapa de ocupación | [0, 0] – [16, 16] m |
| Ancho del robot | 0.497 m |
| Distancia de detección de colisión | 0.001 m |
| Máximo tamaño de arista del árbol de búsqueda | 0.2 m |
| Error máximo permitido (Distancia entre el punto final planificado y el deseado) | 0.02 m |

Tabla . Parámetros escogidos para el planificador.

Pese a que el algoritmo necesita un menor tiempo de ejecución si se escogen valores mayores de tamaño máximo de arista del árbol, se ha decidido elegir un valor de 0.2 m para reducir la posibilidad de que el robot quede atrapado debido a sus restricciones cinemático-dinámicas.

Para comprobar el funcionamiento del planificador, se han generado varias trayectorias a partir de distintos puntos de inicio y de destino. Sin embargo, como se ha explicado en el apartado de control, el sistema de referencia del mapa de ocupación y el del entorno en Apolo no coinciden, por lo que hay que llevar a cabo las transformaciones de coordenadas necesarias con las funciones “*apolo2map*” y “*map2apolo*”.

Los pruebas realizadas para los parámetros escogidos se muestran en la **TABLA**, en la cual se han representado todas las coordenadas en el sistema de referencia del mapa para una mejor visualización.

|  |  |  |  |
| --- | --- | --- | --- |
| Nº Prueba | Pose Inicial [m, m, rad] | Punto final deseado [m, m] | Tiempo de ejecución (s) |
| 1 | [8, 8, pi/2] | [5 10] | 0.04 – 0.07 |
| 2 | [10.7, 0.5, 0] | [5 10] | 0.05 – 0.12 |
| 3 | [10.7, 15.5, 0] | [10.7, 0.5] | 0.13 – 0.19 |

Tabla . Resultados de la planificación de trayectoria.

La representación gráfica de las trayectorias planificadas así como de los árboles de búsqueda resultantes del algoritmo RRT se pueden observar en la *Figura 12*.

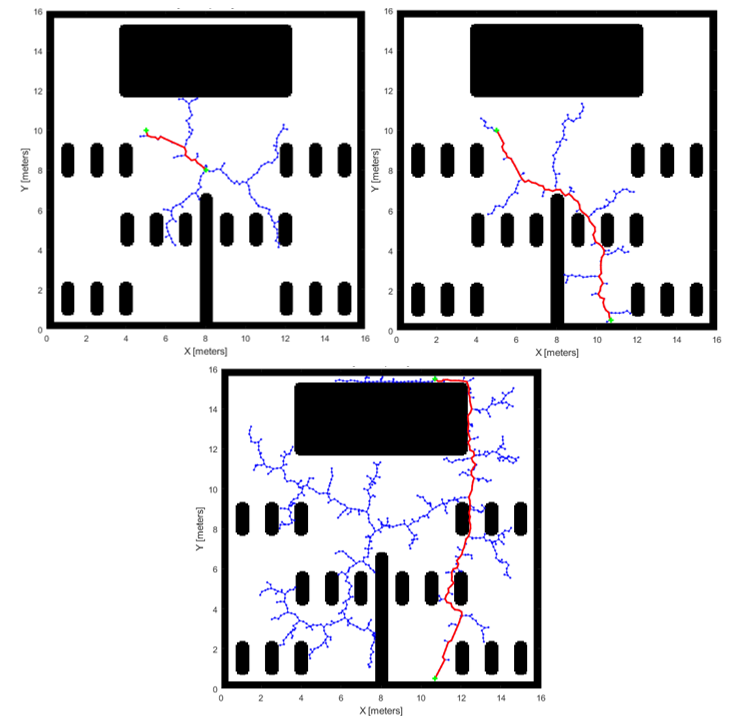


Figura . Trayectorias generadas para las pruebas 1,2 y 3 de la TABLA.

# Demostrador final

Concl