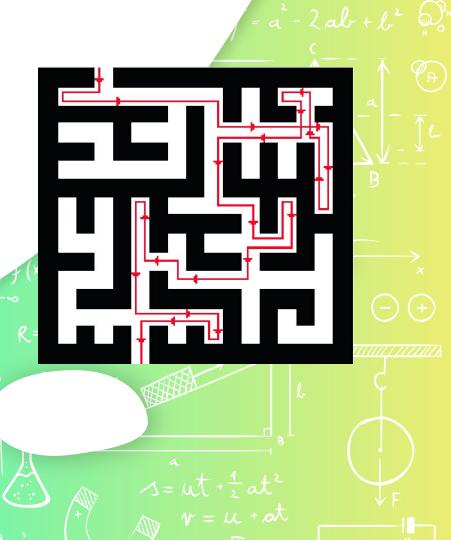
Operating Systems | CETYS Universidad Luis Rodolfo Macias T029806 Gustavo Vazquez T0

Maze Solving Autonomous mobile robot



Maze solving

There are several maze solution algorithms: Dijkstra, A*, Tremaux, Left wall, DFS, BFS and the most popular one Floodfill



Floodfill

It's a basic algorithm for computer graphics, basically it's the representation of how the paint bucket works, but can be modified to work as a maze solving algorithm to work like BFS and DFS.

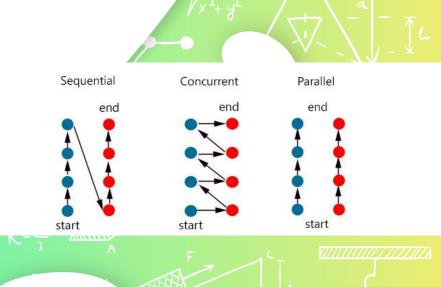
With each iteration of the algorithm, it updates the manhattan distance according to the cell distance.

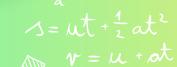
| $T_1 = \ell_1 + 273 = 273 + 60 =$ | | | | | | | | | | | |
|-----------------------------------|---|----------------|---|---|---|-----|---|---|---|----|-------------|
| | | G. | 9 | | 5 | Ci. | 9 | 0 | 9 | | 7 |
| 3 | | G | 9 | 5 | 4 | | | | 9 | | |
| | 9 | (9) | 5 | 4 | | 4 | 5 | | 9 | | |
| | 9 | 5 | 4 | | | | 4 | 5 | 9 | | 4 |
| 3 | 5 | 4 | | | | | | 4 | 5 | 3 | ℓ |
| 3 | 0 | 5 | | | х | | | 5 | 6 | 5 | |
| 8 | 0 | G. | | | | | | 4 | 5 | | 3 |
| 8 | | 04 | | | | | 4 | 5 | 8 | 8 | |
| 8 | | G ₄ | 5 | 4 | | 4 | 5 | 8 | 9 | | 3 3 4 |
| 8 | 9 | 9 | 9 | 5 | 4 | 5 | 9 | 9 | 9 | 22 | |
| 8 | | Ci. | 9 | 9 | 5 | Cs. | 9 | 9 | 9 | S | \bigcap |

Concurrency

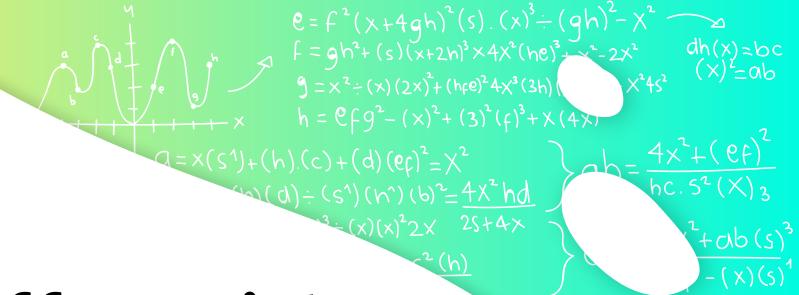
Is the composition of independently executing processes. "Concurrency provides a way to structure a solution to solve a problem that may (but not necessarily) be parallelizable." (Wahome, 2020). (FreeRTOS was used in this implementation to perform

implementation to perform concurrency)



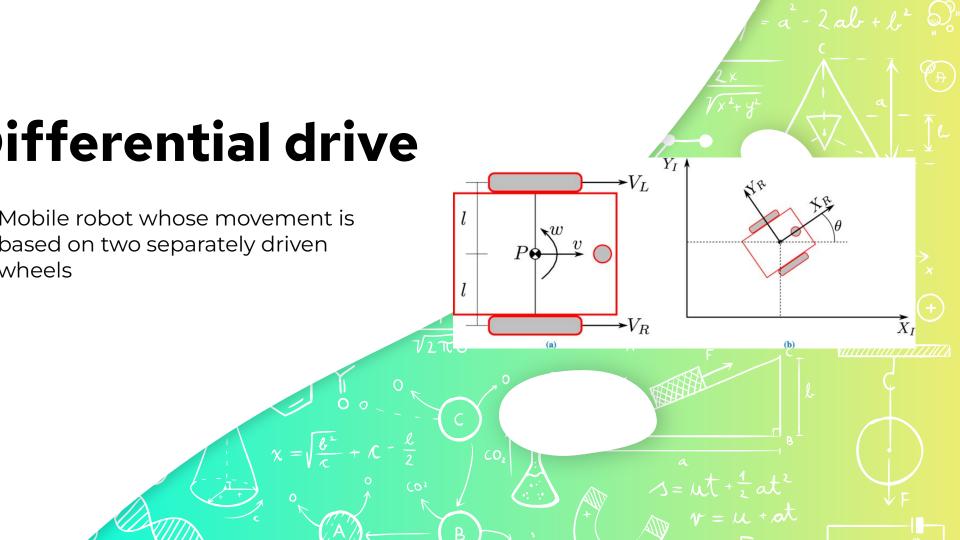






Differential drive Robot





$$\frac{a}{(x)^2 = ab}$$

$$\frac{d}{dx}$$

$$\frac$$

$$e = f^{2}(x+4gh)^{2}(s).(x)^{3}$$

 $f = gh^{2}+(s)(x+2h)^{3}$
 $g = x^{2}+(x)(2x^{2})$ Odol

Odometry

Is the use of data from motion sensors to estimate change in position over time. (Encoders, IMU, etc)

$$Y_R$$

$$Y_R$$

$$Z_{T_L}$$

$$Q_L$$

$$Z_{T_R}$$

$$Q_R$$

$$Z_{T_R}$$

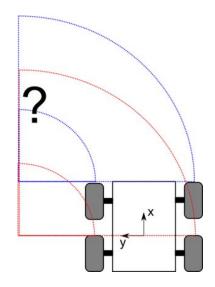
$$Q_R$$

$$Z_{T_R}$$

$$x(t+dt) = x(t) + R \frac{\Delta \varphi_R + \Delta \varphi_L}{2} \cos \theta(t)$$

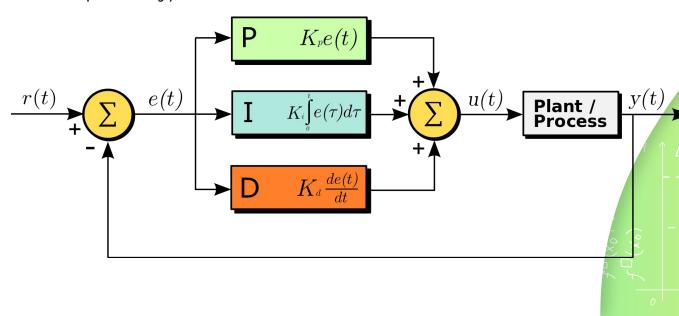
$$y(t+dt) = y(t) + R \frac{\Delta \varphi_R + \Delta \varphi_L}{2} \sin \theta(t)$$

$$\theta = R \frac{\varphi_R - \varphi_L}{2b}$$

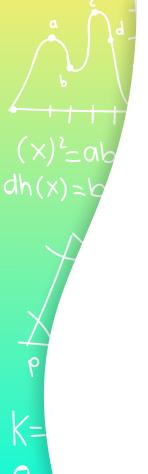


PID Controller

A PID controller continuously calculates an error value e(t) as the difference between a desired value and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively).



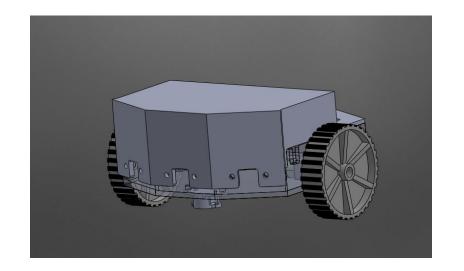


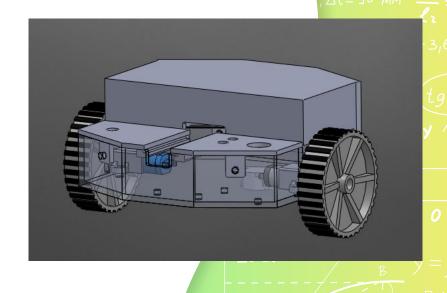


Implementation



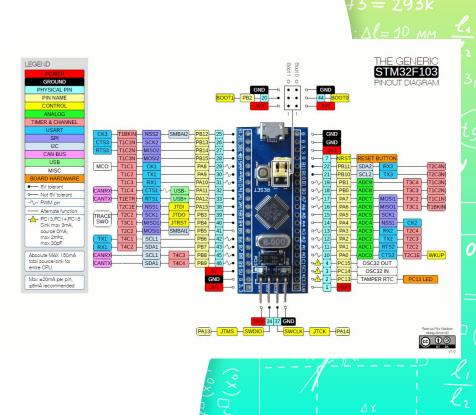
Mechanical design





Electronics

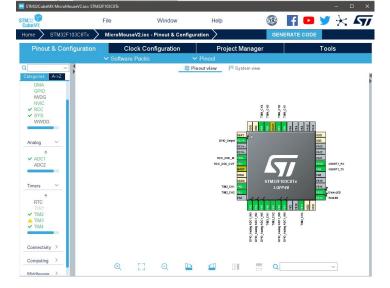
- TCR500
- GM12-GAN20 Motors w/Encoders
- STM32F130C8T6 (AKA Blue PIII)
 - 4 Timers (Encoders, ADC, UART, I2C, etc.)
 - o 10 ADC pins
 - Hardware solution for encoder reading
 - o ADC -> DMA
- LiPo Battery 3.7 V
- XI6009E1 Step Up to 3.5-30 V
- DRV8833 Motor driver



Software

For the software part, we used STM32CubeMX for the initial configuration of all the pins, timers, control structure, etc. And for the software uploading and editing we used STM32CubeIDE. As a middleware to help with concurrency tasks FreeRTOS was

used.



```
Project - MicroMouseV2/Core/Src/main.c - STM32CubelDE
         Source Refactor Navigate Search Project Run Window Help
             □ + ~ ~ ~ ~ ~ ~ | → | 
Project Explorer X
                             c main.c X c main.c
                                                     c main.c
                                                                              c system_stm32f1xx.c
                              712 /* USER CODE END Header StartOdometryUpdate */
                              713@ void StartOdometryUpdate(void const * argument)
> IDE BlinkBluePill (in STM32CubelD
                              714 {
> IDE Blinky
                                     /* USER CODE BEGIN 5 */
                              715
> IDE CB (in STM32CubeIDE)
                                     /* Infinite loop */
                                     for(;;)
> WE CircularBuffer Class
                              718
    EncodeBluePillTest
                              719
                                         RTicks = ((TIM2->CNT)>>2);
     EncoderMotorController
                              720
                                             LTicks = ((TIM3->CNT)>>2);
     GoalFollowerMM
> IDE Interrupt LEDS
                              722
                                             //CNT registers are 2^14
> IDE Interrupt_LEDS_2
                              724
                                             if (RTicks >= 16000 && PrevRTicks <= 1000) {
    Interrupt LEDS 3
                              725
                                                 RTicks = -(16383 - RTicks):
    MicroMouseV1
                              726
> WE MicroMouseV2
                              727
> DE PathFollowingMM
                              728
                                             if (LTicks >= 16000 && PrevLTicks <= 1000) {
                                                 LTicks = -(16383 - LTicks);
                              729
> DE PathTrackingMM
                              730
> WE Periodict task
                              731
> IDE PPM (in STM32CubeIDE)
                                             if (RTicks <= 1000 && PrevRTicks >= 16000) {
                              732
> IDE PTL_Length
                                                 RTicks = RTicks + (16383 - PrevRTicks);
> UE UART TX
                              734
                              735
    UART TX CIRCULAR BUFFER
                              736
                                             if (LTicks <= 1000 && PrevLTicks >= 16000) {
> IDE UART_TX_CIRCULAR_BUFFER_1
                                                 LTicks = LTicks + (16383 - PrevLTicks):
    UART_TX_CIRCULAR_BUFFER_
                              738
> IDE UART_TX_CIRCULAR_BUFFER_
                              739
 UART TX Interrupt
                              740
                                             RdtTicks = RTicks - PrevRTicks:
                              741
                                             LdtTicks = LTicks - PrevLTicks:
                              742
                              743
                                             Rdist = mmPerTick*RdtTicks;
                              744
                                             Ldist = 2*mmPerTick*LdtTicks: //Resolution of left encoder is
                              745
                                             Cdist = (Rdist + Ldist)/2;
                              746
                              747
                                             x = x + Cdist*cos(theta);
                              748
                                             y = y + Cdist*sin(theta);
                              749
                              750
                                             theta = theta + ((Rdist - Ldist)/L);
                              751
                                             theta = atan2(sin(theta), cos(theta));
                              752
                                             thetaDeg = theta*(180/3.1416);
                              753
                                           Writable
                                                               Smart Insert
                                                                                 123:36 [35]
expected expression before '=' token
```