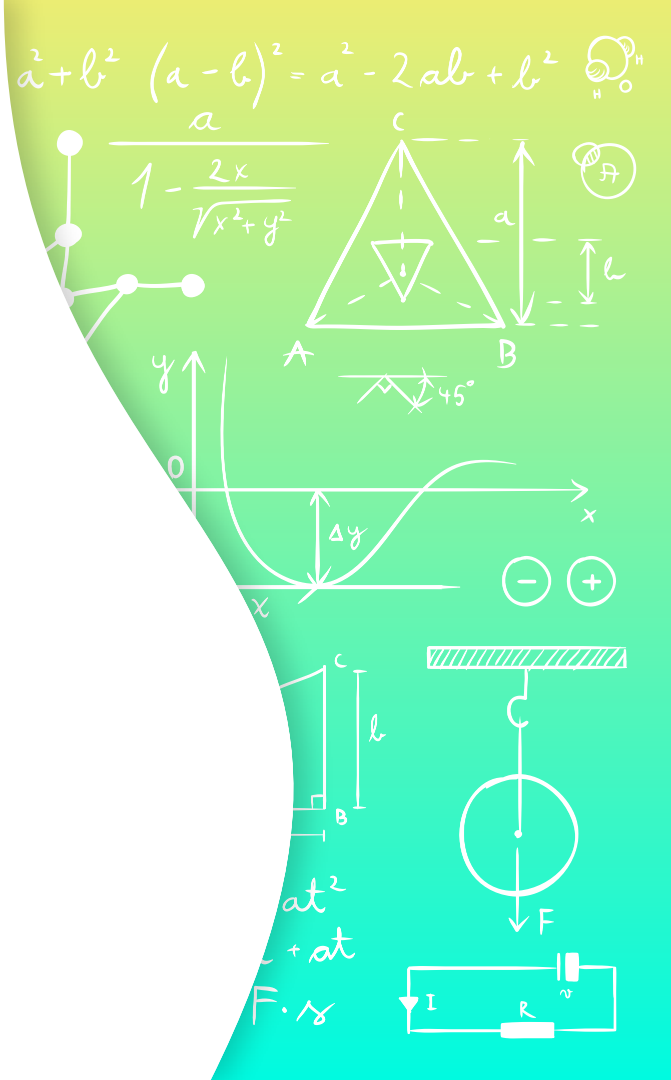
Operating Systems | CETYS Universidad

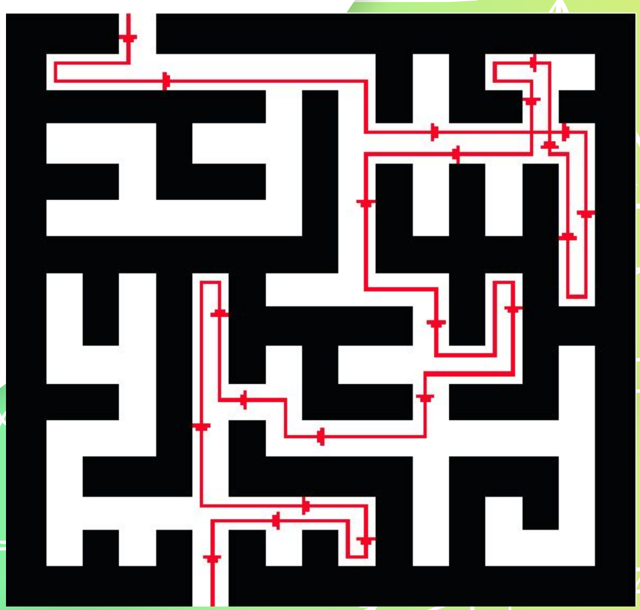
Luis Rodolfo Macias T029806
Gustavo Vazquez T0

Maze Solving Autonomous mobile robot



Maze solving

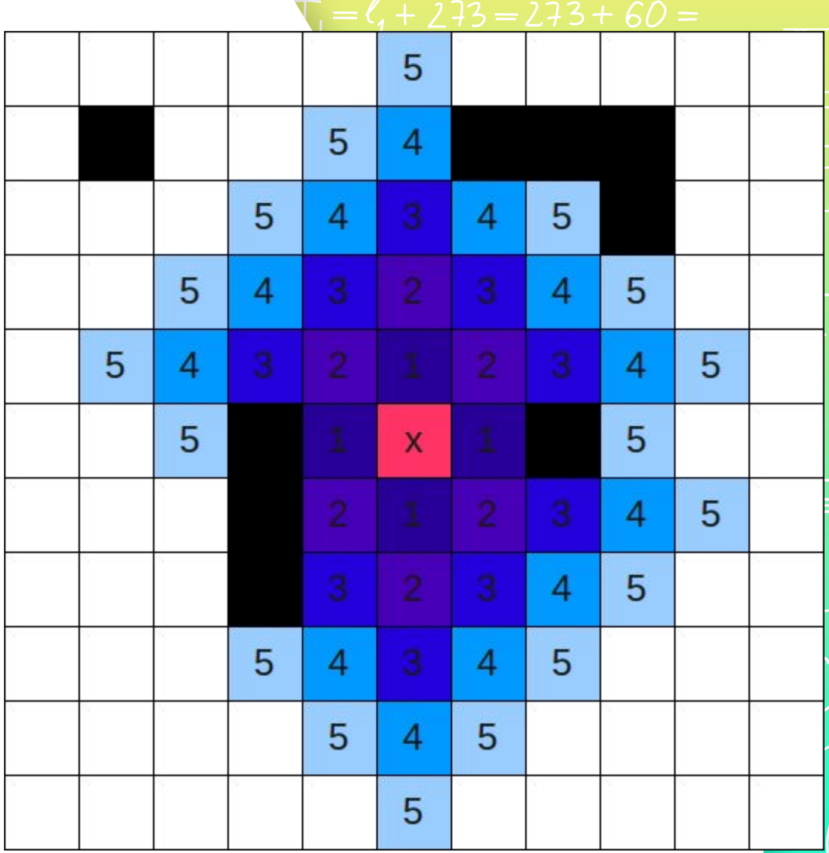
There are several maze solution algorithms: Dijkstra, A*, Tremaux, Left wall, DFS, BFS and the most popular one Floodfill



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.

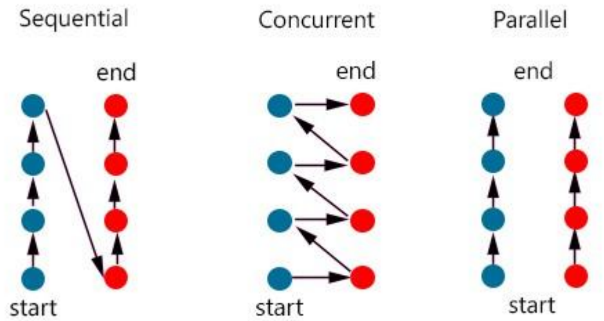
With each iteration of the algorithm, it updates the manhattan distance according to the cell distance.

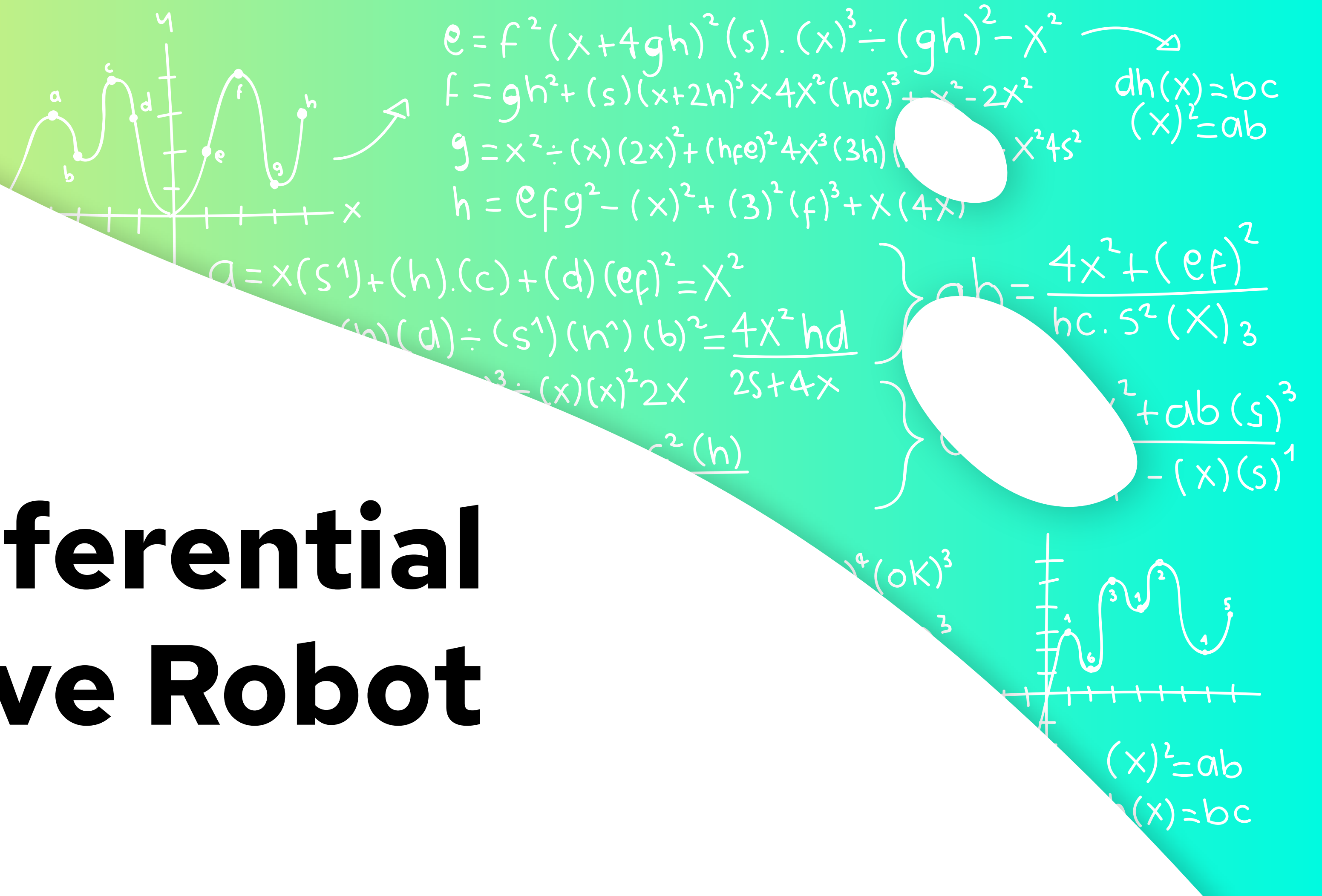


					5					
	■			5	4	■	■	■		
			5	4	3	4	5	■		
		5	4	3	2	3	4	5		
	5	4	3	2	1	2	3	4	5	
		5	■	1	x	1	■	5		
			■	2	1	2	3	4	5	
			■	3	2	3	4	5		
			5	4	3	4	5			
				5	4	5				
					5					

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 concurrency)

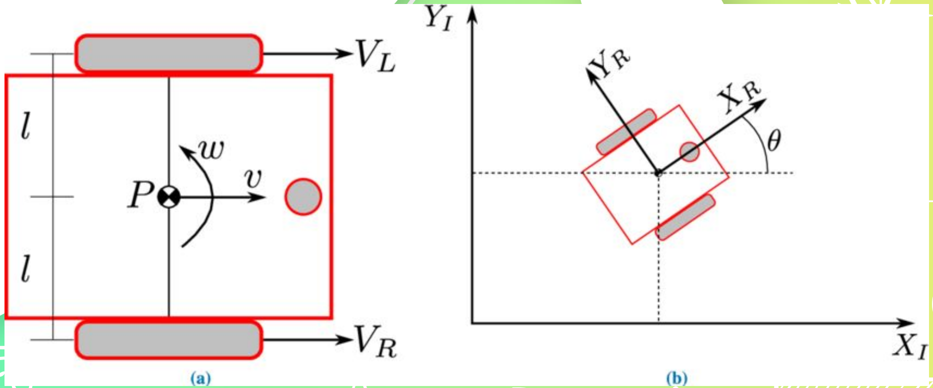




Differential drive Robot

Differential drive

Mobile robot whose movement is based on two separately driven wheels

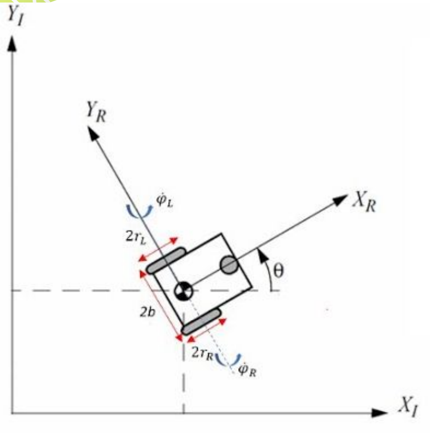


(a)

(b)

Odometry

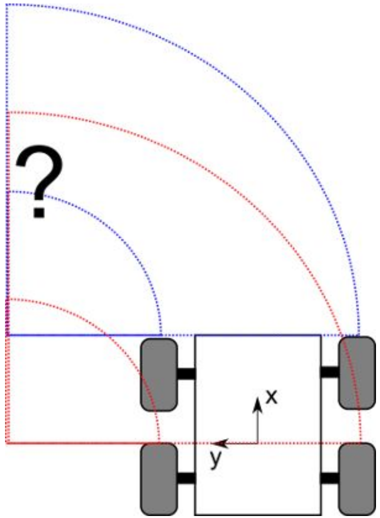
Is the use of data from motion sensors to estimate change in position over time. (Encoders, IMU, etc)



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

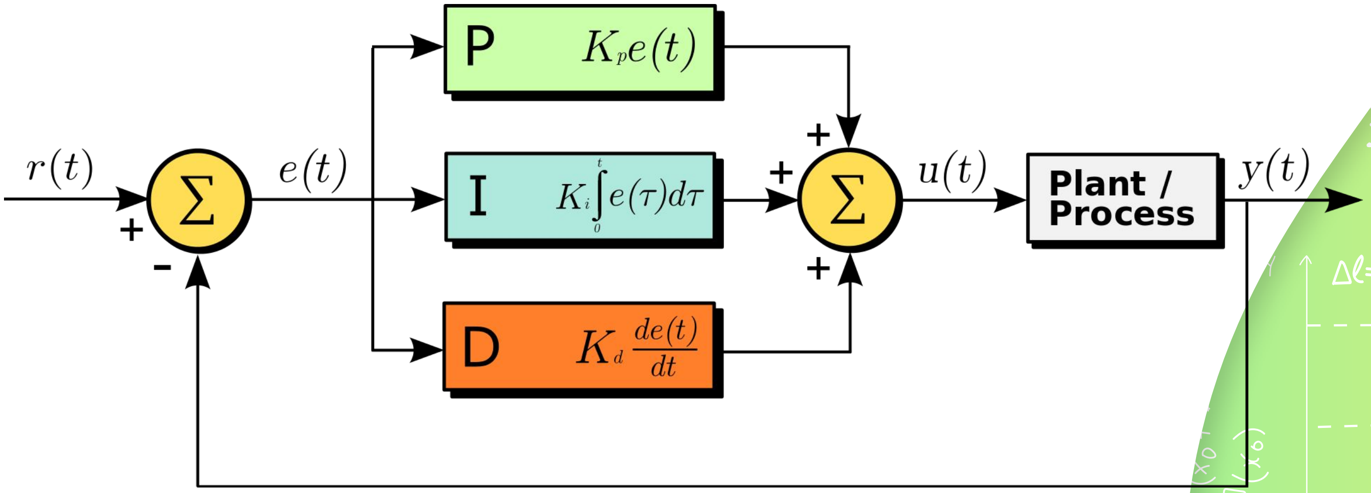
$$y(t+dt) = y(t) + R\frac{\Delta\varphi_R + \Delta\varphi_L}{2}\sin\theta(t)$$

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



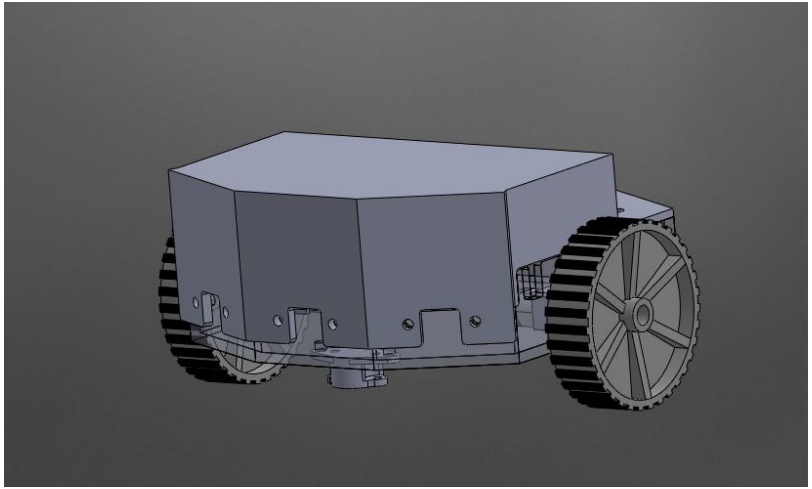
PID Controller

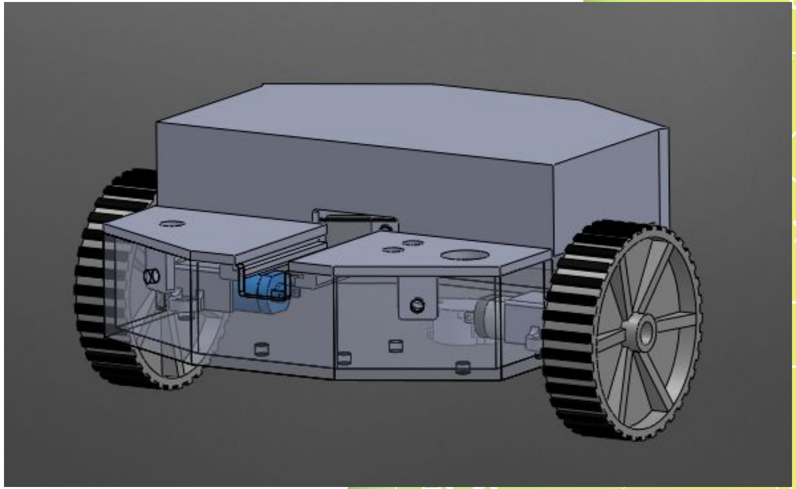
A PID controller continuously calculates an error value e(t) as the difference between a desired value and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively).



Implementation

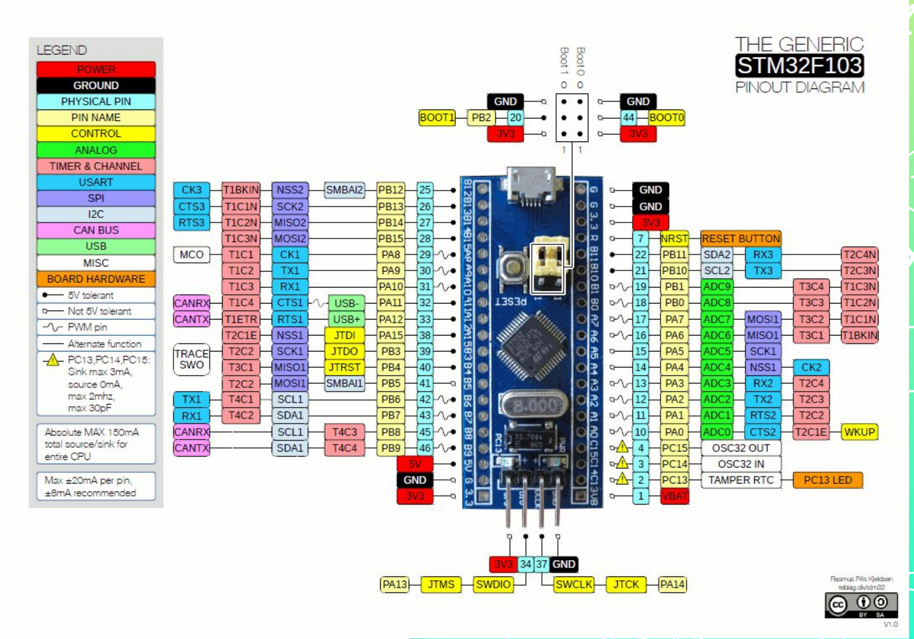
Mechanical design





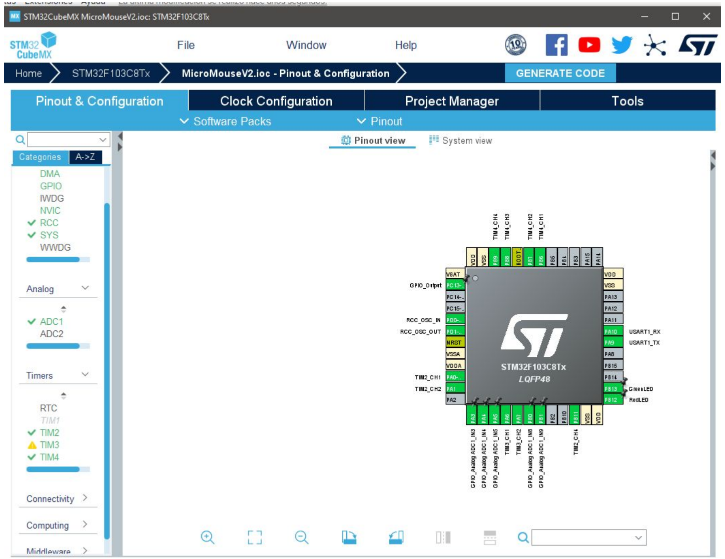
Electronics

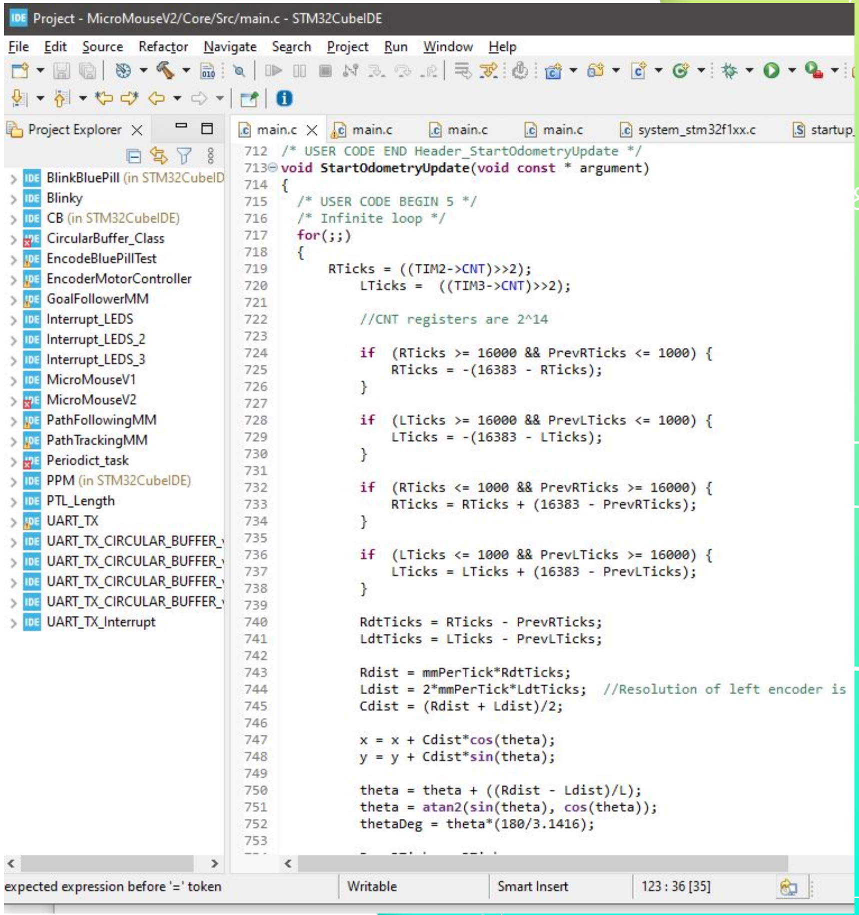
- TCR500
- GM12-GAN20 Motors w/Encoders
- STM32F130C8T6 (AKA Blue PIII)
  - 4 Timers (Encoders, ADC, UART, I2C, etc.)
  - 10 ADC pins
  - Hardware solution for encoder reading
  - ADC -> DMA
- LiPo Battery 3.7 V
- XI6009E1 Step Up to 3.5-30 V
- DRV8833 Motor driver



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.





```c
/* USER CODE END Header_StartOdometryUpdate */
void StartOdometryUpdate(void const * argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
      RTicks = ((TIM2->CNT)>>2);
          LTicks =  ((TIM3->CNT)>>2);

          //CNT registers are 2^14

          if  (RTicks >= 16000 && PrevRTicks <= 1000) {
              RTicks = -(16383 - RTicks);
          }

          if  (LTicks >= 16000 && PrevLTicks <= 1000) {
              LTicks = -(16383 - LTicks);
          }

          if  (RTicks <= 1000 && PrevRTicks >= 16000) {
              RTicks = RTicks + (16383 - PrevRTicks);
          }

          if  (LTicks <= 1000 && PrevLTicks >= 16000) {
              LTicks = LTicks + (16383 - PrevLTicks);
          }

          RdtTicks = RTicks - PrevRTicks;
          LdtTicks = LTicks - PrevLTicks;

          Rdist = mmPerTick*RdtTicks;
          Ldist = 2*mmPerTick*LdtTicks;  //Resolution of left encoder is
          Cdist = (Rdist + Ldist)/2;

          x = x + Cdist*cos(theta);
          y = y + Cdist*sin(theta);

          theta = theta + ((Rdist - Ldist)/L);
          theta = atan2(sin(theta), cos(theta));
          thetaDeg = theta*(180/3.1416);
```

expected expression before '=' token