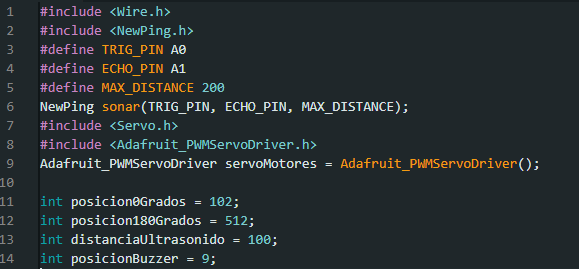
Definición de librerías y variables



Definimos el sensor ultrasonido

Texto

Descripción generada automáticamente

Llamamos los métodos posicioninicial

Espera unos 1000 milisegundos

Después llama el método moveforward

Declaramos las distancias a la derecha

Y a la izquierda

Verificamos que la distancia que tome el sensor

Ultrasonido

Sea menor a 20 y mayor 0 cm si esto se cumple

Sonara el buzzer

Y el robot se detendrá y esperará 200 milisegundos

Definimos la posición 0 grados

Y la posición 180 grados para los servos

Definimos la variable distancia

Definimos la variable buzzer

Activamos los servomotores

A una frecuencia de 50 Hz

Lee la distancia del ultrasonido cada 100 milisegundos

Definimos las variables del ultrasonido

 Texto

Descripción generada automáticamente

Verificamos si la distancia de la derecha es mayor a la izquierda

Para girar a la derecha

Si no se ejecutara el giro a ala izquierda

Y si ninguna de las dos se cumple seguira hacia adelante

Y leera la distancia

Llamamos los métodos posicioninicial

Después llama el método moveforward

Declaramos las distancias a la derecha

Y a la izquierda para luego leerlas

Verificamos que la distancia que tome el sensor Ultrasonido

Sea menor a 20 y mayor 0 cm si esto se cumple

Sonara el buzzer

Y el robot se detendrá y retrocederá y esperará 200 milisegundos

después la cabeza girara 180 grados a la derecha y después a la izquierda

Se desactivará el sonido del buzzer

Definimos la posición 0 grados

Y la posición 180 grados para los servos

Definimos la variable distancia

Definimos la variable buzzer

Activamos los servomotores

A una frecuencia de 50 Hz

Lee la distancia del ultrasonido cada 100 milisegundos

Definimos en las variables del ultrasonido

Texto

Descripción generada automáticamente

después retrocederá y esperará 200 milisegundos

 volverá a detenerse y esperará 2000 milisegundos

Y activa la función de mirar a la derecha e izquierda

Y por último desactiva el sonido del buzzer

Verificamos que la distancia de la derecha sea mayor o igual a la izquierda

Si cumple girará ala derecha y se detendrá

Si no

Girará ala izquierda y se detendrá

Si no se cumple las dos condiciones se moverá hacia adelante y leera la distancia

Definimos el método

Texto

Descripción generada automáticamente

El servo motor del cuello girara 50 grados

Hacia a la derecha

Leera la distancia del ultrasonido

El servo motor volverá a su posición inicial

El servo motor del cuello girara 180 grados

Hacia a la izquierda

Leera la distancia del ultrasonido

El servo motor volverá a su posición inicial

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Definimos un atributo del sensor ultrasonido

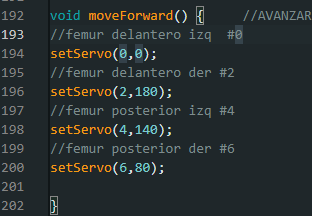
La distancia la convertimos a centímetros

Declaramos un método

Para la posición inicial del Robot cuadrúpedo

Cada servo se ajusta a 0, 90 ,180 grados

Método para avanzar



Método para Detenerse

Texto

Descripción generada automáticamente con confianza baja

Cada servo se ajusta a 0, 90 ,180 grados

Cada servo se ajusta a 0, 90 ,180 grados

Método para retroceder

Texto

Descripción generada automáticamente

Método para girar a la derecha

Texto

Descripción generada automáticamente

Cada servo se ajusta a 0, 90 ,180 grados

Cada servo se ajusta a 0, 90 ,180 grados

Método para girar ala izquierda

Texto

Descripción generada automáticamente

Método para activar el buzzer

Texto

Descripción generada automáticamente

Método para detener el sonido

Del buzzer

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Declaramos un método setServo

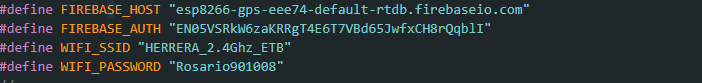
para hacer la conversión a grados de los atributos pos0, pos 180 de los servomotores

Importamos Librerías

Texto

Descripción generada automáticamente

Definimos el host de la base de datos de Firebase y su respectivo token





Variables de conexión del GPS

Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Incluir en Arduino la librería ArduinoJson versión v5.13.5

Tambien agregamos la SSID de nuestra red y su PASSWORD

Comenzamos la conexión del firebase con el Esp8266

Activa el GPS y manda al firebase la longitude y la latitude

Si no se cumple lo anterior nos arroja error

Y espera 5 segundos para volver a intentar

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Método para poder leer los datos del GPS-neo-6m

Método llamado al principio para saber si esta conectado el esp8266 al internet

Método para imprimir si es correcta la conexión del Firebase.

Método para imprimir si la conexión es correcta con el Firebase.

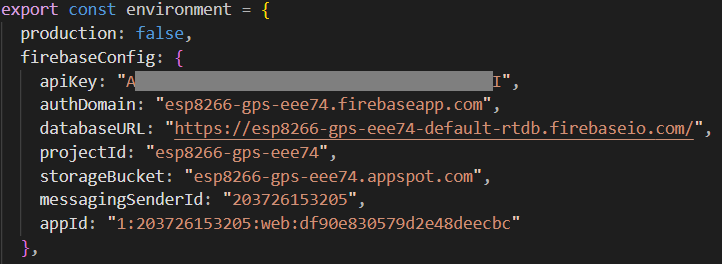
Método para volver a intentar la conexión con Firebase volviendo a pasar los parámetros como el host y el token.

**Documentación código aplicativo móvil**

Se crea el proyecto en Ionic, haciendo uso de Angular.

Es importante colocar el comando “npm install --legacy-peer-deps” para instalar todas las librerías correspondientes, entre ellas destacan

* AngularFire
* Angular Google Maps





Texto

Descripción generada automáticamente

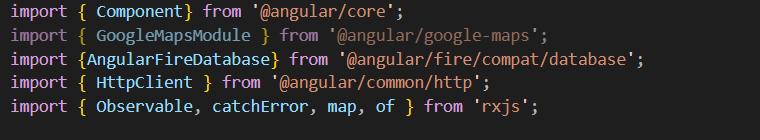
Para encontrar los archivos de la aplicación en Ionic, es necesario ir a la carpeta ***src***, allí encontrará todas las carpetas necesarias tanto de la aplicación como de ambientes para configurar.

En la carpeta de **src/enviroments** tanto para el archivo ***enviroment.prod.ts*** y ***enviroment.ts*** se establece la configuración de la API de Firebase con su respectiva key.

Se importan las siguientes librerías del paquete AngularFire y Google Maps instalado en el archivo ***app.module.ts.***

Una vez importados, en ***app.modules.ts***, en la sección de @NgModule se importan también, esto con el fin de poder interactuar con la base de datos de Firebase y que el mapa pueda visualizarse en la aplicación.

.





Texto

Descripción generada automáticamente

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

En la carpeta de ***src/app/home/home.page.ts*** se importan las librerías completas instaladas anteriormente

Dentro de la clase HomePage, dentro del constructor se crea el valor de la base de datos haciendo referencia a la librería de Angular con Firebase y a otra librería llamada HttpClient la cual permite hacer peticiones.

Luego se crean dos variables, *latitude* y *longitude* que son aquellas que tomarán las coordenadas del robot cuadrúpedo y las mostrarán en el mapa. La variable apiLoaded funcionará para detectar si el mapa se creó correctamente.

Las variables markerPosition, center, markerOptions y zoom son encargadas de posicionar el mapa y establecerlo con un aumento predeterminado, junto con algunas coordenadas y opciones que le dan más orden cuando se genere.

Seguidamente, se crea una constante que toma hace referencia a el nombre de la base de datos, para luego buscar los valores dentro de Firebase para guardarlos en gpsData.

Finalmente, el condicional verificará si hay información dentro y si hay, que asigne los valores correspondientes a la latitud y longitud para que sean mostrados en el mapa, creando el marcador y dirigiéndose a esas coordenadas

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

apiLoaded se encargará de verificar si el API carga correctamente el mapa mediante un método HTTP y un pipe, en caso de no hacerlo y que falle el metodo, mostrará un error.

Para finalizar, se coloca la etiqueta HTML del mapa de Google con los parámetros correspondientes a las variables creadas en **home.page.ts** como lo es la ubicación del mapa y el zoom, sumado a la posición del marcador que se creará eventualmente.