UNIVERSIDAD DE GRANADA

Robótica Móvil

Guiado GPS del robot ZUMO 32U

Jaime Lozano Tortosa
jaimelozano@ugr.es

Índice de contenidos

1. Antecedentes y objeto

Uno de los grandes problemas de la localización es la obtención de la posición absoluta. Podemos tener un mapa muy preciso del entorno en el que vamos a trabajar, pero si no conocemos la posición sobre ese mapa no es posible emplear esos datos.

La tecnología GPS permite ubicar de forma absoluta un robot situado en exteriores. La precisión depende en gran medida de la visibilidad de los satélites y del tipo de tecnología empleada, y va de unos 10m hasta centímetros. En este trabajo se pretende realizar un control de la dirección por GPS.

2. Trabajo realizado

El control de la dirección se ha realizado de la siguiente manera.

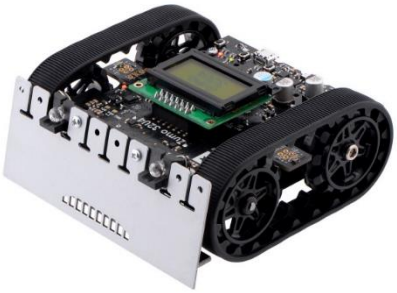
En primer lugar, se introducen las coordenadas del punto objetivo en latitud y longitud. Posteriormente, se obtiene la posición en latitud y longitud del robot.

En segundo lugar, ambas coordenadas se pasan a coordenadas cartesianas. Posteriormente, se restan término a término, obteniendo un vector que indica la dirección y la distancia al objetivo.

En tercer lugar, se emplea una brújula para conocer el azimut del robot. Con este azimut ya podemos dirigir la marcha hacia el objetivo.

Se han empleado los siguientes componentes:

- PMOD GPS: es un receptor GPS que incorpora la antena.
- Raspberry Pi Zero W: es la versión Raspberry más pequeña. Su procesador es de 1GHz y 500mb de RAM. Se ha instalado raspbian sin entorno gráfico.
- ZUMO32U



1. Parseo de tramas NMEA
2. Paso de WGS84 a cartesianas
3. Introduzco coordenadas XY del objetivo
4. Obtención de ángulo y distancia al objetivo
5. Paso ángulo por serie al ZUMO

1. Recibo por serie el ángulo a objetivo
2. Con el magnetómetro calculo el ángulo al norte
3. Calculo el ángulo a corregir
4. Empleo el giróscopo para saber el ángulo corregido

En la imagen superior se pueden ver los 3 componentes que integran el sistema. El receptor se ha conectado por serie con la raspberry y ésta se ha conectado por USB al robot ZUMO.

Se ha desarrollado un programa en Python que parsea las tramas NMEA obteniendo el ángulo al objetivo. Este programa corre en la raspberry.

En el robot ZUMO se ha desarrollado el programa que emplea el magnetómetro y el giróscopo para apuntar hacia el objetivo. Se ha desarrollado además la librera MagSensor para hacer uso del magnetómetro.

3. Problemática

Durante el desarrollo del proyecto ha surgido un contratiempo. La brújula que se encuentra instalada en el robot está situada en la pcb muy próxima a los motores. Los motores están compuestos por imanes que cambian completamente los valores de la brújula. Se ha probado con el empleo de filtros, pero ha sido imposible realizar un control adecuado de la dirección.

4. Soluciones

Algunas alternativas a la falta de la brújula pueden ser el empleo del giróscopo. Este sensor nos indica la velocidad angular, integrando se obtiene la posición angular o azimut. Iniciando el programa apuntando al norte permite conocer el azimut del robot. El problema de esta solución es que existe una deriva. De forma experimental se ha observado que la deriva es de +-1º por minuto. Es la solución que se ha empleado en este trabajo.

Otra posible solución sería emplear el mismo GPS. Comenzando en una dirección aleatoria y viendo si la distancia al objetivo aumenta o disminuye es posible conocer la dirección. Esta aproximación es muy teórica y que dada la poca precisión del receptor empleado la dirección recorrida para conocer el ángulo es excesivamente grande.

5. Conclusiones

La tecnología GPS permite dar solución de posición absoluta de forma rápida, precisa y con bajo coste. En aquellas situaciones en las que no se puede emplear odometría como aviación o navegación marítima, es posible obtener fácilmente la posición con un receptor GPS.

Por otro lado, el uso de GPS en combinación con sensores inerciales permite hacer el sistema más robusto ante una posible pérdida de la señal GPS.

La plataforma ZUMO32U es una potente herramienta ya que permite centrar el esfuerzo en el desarrollo del control. Además, cuenta con numerosas librerías y ejemplos. Para este trabajo han sido de gran ayuda los ejemplos MazeSolver y RotationResist.

6. Bibliografía

https://forum.arduino.cc/index.php?topic=508325.0
https://www.blascarr.com/joystick-arduino-orientation/
https://sciencing.com/calculate-antipode-6170267.html

Anexo I: código del robot ZUMO 32U

```
#include <Zumo32U4.h>
#include "TurnSensor.h"
#include "MagSensor.h"

Zumo32U4Motors motors;
Zumo32U4IMU imu;

int incomingByte = 0;
int speed = 100;
int32_t last_t;

int angulo_a_norte;
int angulo_a_objetivo;
int angulo_a_corregir;
int angulo_corregido;

void setup()
{
  Serial.begin(9600);
  turnSensorSetup();
  magSensorSetup();
  delay(1000);

  angulo_a_norte = 0;
  angulo_a_objetivo = 0;
}

void loop()
{
  if(Serial.available()){
    angulo_a_objetivo = Serial.parseInt();
    while(Serial.available()) Serial.read();
    angulo_a_corregir = angulo_a_norte - angulo_a_objetivo;
  }

  if (angulo_corregido < angulo_a_corregir){
    while (angulo_corregido < angulo_a_corregir){
      motors.setSpeeds(-speed,speed);
      turnSensorUpdate();
      angulo_corregido = (((int32_t)turnAngle >> 16) * 360) >> 16;
    }
    motors.setSpeeds(0,0);
  }
  else if (angulo_corregido > angulo_a_corregir){
    while (angulo_corregido > angulo_a_corregir){
      motors.setSpeeds(speed,-speed);
      turnSensorUpdate();
      angulo_corregido = (((int32_t)turnAngle >> 16) * 360) >> 16;
    }
    motors.setSpeeds(0,0);
  }
}}
```

Anexo II: librería MagSensor.cpp

```
#include "MagSensor.h"
#include "TurnSensor.h"
#include <Zumo32U4.h>
#include <Wire.h>
#include <LIS3MDL.h>

LIS3MDL mag;
Zumo32U4LCD lcd;
extern Zumo32U4Motors motors;


int mx_max, mx_min, my_max, my_min;


void magSensorSetup(){
  float mx,my;

  Wire.begin();
  mag.init();
  mag.enableDefault();

  lcd.clear();
  lcd.gotoXY(0,0);
  lcd.print(F("Mag cal"));

  mag.read();
  mx_max = mag.m.x;
  my_max = mag.m.y;
  mx_min = mag.m.x;
  my_min = mag.m.y;

  while (angulo_corregido != -180){
    motors.setSpeeds(-speed,speed);
    mag.read();
    mx = mag.m.x;
    my = mag.m.y;
    angulo_corregido = (((int32_t)turnAngle >> 16) * 360) >> 16;
    if (mx > mx_max){
      mx_max = mx;
    }
    if (my > my_max){
      my_max = my;
    }
    if (mx < mx_min){
      mx_min = mx;
    }
    if (my < my_min){
      my_min = my;
    }
    turnSensorUpdate();
  }
  while (angulo_corregido != 0){
    motors.setSpeeds(-speed,speed);
    turnSensorUpdate();
    mag.read();
    mx = mag.m.x;
    my = mag.m.y;
    angulo_corregido = (((int32_t)turnAngle >> 16) * 360) >> 16;
    if (mx > mx_max){
```

```
      mx_max = mx;
    }
    if (my > my_max){
      my_max = my;
    }
    if (mx < mx_min){
      mx_min = mx;
    }
    if (my < my_min){
      my_min = my;
    }
  }
  motors.setSpeeds(0,0);
  lcd.clear();
}

void magSensorUpdate(){
  float mx,my;
  mag.read();
  mx = mag.m.x;
  my = mag.m.y;
  mx = map(mag.m.x,mx_min,mx_max,32767,-32768);
  my = map(mag.m.y,my_min,my_max,32767,-32768);
  angulo_a_norte = atan2(my,mx)*180/(2*PI);
  angulo_a_norte = map(angulo_a_norte,-90,90,180,-180);
}
```

Anexo III: librería MagSensor.h

```
extern int angulo_a_norte;
extern int angulo_corregido;
extern int speed;

void magSensorSetup();
void magSensorUpdate();
```

Anexo IV: programa Python

```
import io
import time
import pynmea2
import serial
import utm
import math

ser = serial.Serial('/dev/ttyS0', 9600, timeout=1.0)
sio = io.TextIOWrapper(io.BufferedRWPair(ser, ser))

ser2 = serial.Serial('/dev/ttyACM0', 9600, timeout=1.0)

while 1:
    try:
        line = sio.readline()
        msg = pynmea2.parse(line)
        if (msg.sentence_type == 'RMC'):
                print(msg.timestamp)
                input_lat = msg.latitude
                input_lon = msg.longitude
                x, y, zone_number, zone_letter  =
utm.from_latlon(input_lat,input_lon)
                # Calcular error
                input_lat0 = 37.18817 # Granada
                input_lon0 = -3.60667 # Granada
#                input_lat0 = 48.85341 # Paris
#                input_lon0 = 2.3488   # Paris
#                input_lat0 = 40.4165 # Madrid
#                input_lon0 = -3.70256 # Madrid
                x0, y0, zone_number0, zone_letter0 =
utm.from_latlon(input_lat0,input_lon0)
                xerror = x0 - x
                yerror = y0 - y

                distance = math.sqrt(xerror*xerror + yerror*yerror)
                print(str('Distance: ') + str(int(distance)))

                angulo_a_objetivo = math.atan2(xerror,yerror)*180/3.1415
                print(str('Angle: ') + str(int(angulo_a_objetivo)))

                # Pasar angulo_a_objetivo
                command = str(angulo_a_objetivo) + str('\n')
                ser2.write(bytes(command, encoding='utf8'))

    except serial.SerialException as e:
        print('Device error: {}'.format(e))
        break
    except pynmea2.ParseError as e:
        print('Parse error: {}'.format(e))
        continue
    except AttributeError as e:
        print('Atrribute error: {}'.format(e))
        continue
```