Pujada en Costa

Projectes de programació 1

By

Alex Almansa, Marc Llort

Departament d'enginyeria

La Salle URL

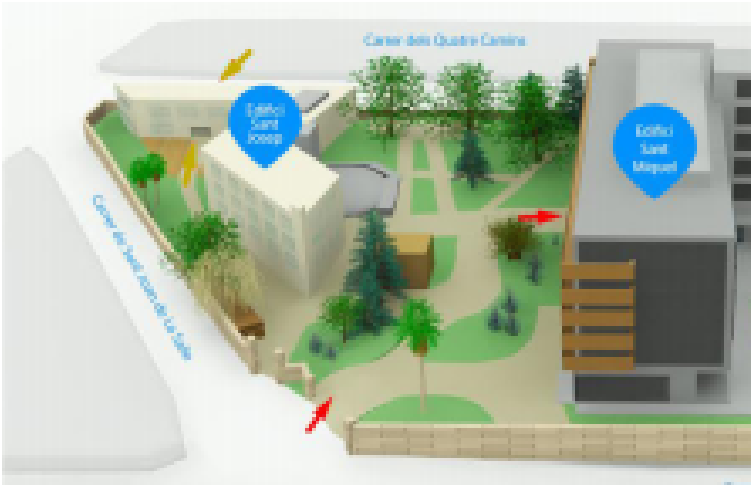
Desembre 2018

Índex

1 Introducció

Pujada en costa és una competició anual on els LsMaker han de fer una de les pujades del parc de la universitat, per intentar arribar els primers a la part més alta.
Per dur a terme el objectiu, podrem dur a terme tot tipus de modificacions tant de software com de hardware, però sempre tenint en compte que ha de ser un robot totalment autònom, per tant nosaltres només podrem indicar-li quan ha de començar a moure's, res més.

També és important recalcar el fet de que durant el inici de la carrera, podrem tenir el robot col·locat en qualsevol direcció, per tant ja des de un inici haurà d'anar calculant ell sol cap on és el punt més elevat del circuit per així poder encarar-lo i començar el seu viatge cap a la meta.



2 Software

En aquest apartat explicarem les diferents modificacions de software que hem fet al projecte base que teníem del lsMaker per fer-lo autònom i ràpid per a que pugui arribar el primer a la línia de meta.
El primer pas va ser analitzar el funcionament del projecte quan funcionava amb el mòbil com a comandament remot.
A partir d'aquí, vam poder veure que havia dos formes de dur a terme el nostre objectiu. La primera es tractava de fer us de les funcions LsMtRecte Dreta Esquerra, passant diferents variables de graus, velocitat inicial i final etc. Després de moltes hores de intentar entendre correctament el funcionament d'aquestes funcions, i de que de vegades ens funcionessin correctament i d'altres no, vam decidir intentar fer-ho d'alguna forma diferent.
La segona opció, i la que finalment vam acabar utilitzant, era la de fer servir les funcions: LsMandoTuneig i LsMtMando. D'aquesta manera el que fem es que simulem com si hagués un mando, que en aquest cas és el nostre programa, i aquest els hi va passant la acció que volem realitzar a LsMandoTuneig, la qual adecua les dades per LsMtMando.
Un cop les dades estan correctament adequades, executem la funció LsMtMotor, el que provoca que el LsMaker es mogui segons el que hem programat.

2.1 Inicialització

En un principi, al encendre el robot, s'executa la funció setup(), on es mira si cal calibrar, encén la pantalla i es queda esperant a la funció aciloop().
Aquesta funció es la que rep les instruccions del mòbil. Nosaltres la hem modificat perquè tal i com rebi qualsevol instrucció, la ignori sigui la que sigui, però faci que el robot comenci a funcionar.
Aquest çomençar a funcionar"serà quan surti de la funció setup() i entri a loop(), on executarà la funció magia(), la qual explicarem posteriorment, i la funció LSperiferics() per anar actualitzant les dades de l'acceleròmetre. Es important que s'actualitzin perquè ens han sigut molt útils per poder debuggar correctament el funcionament del robot.
Dins de LsPerifèrics, hem fet una petita modificació perquè en comptes d'actualitzar la informació cada 2.5s, ho faci cada cop que s'executi, d'aquesta manera el nostre robot detectarà els canvis de pendent de forma més ràpida i precisa.

2.2 AI

Hem creat la nostra funció magia(), la qual s'encarrega de controlar el robot per a que busqui i es mogui en direcció de la major pendent.

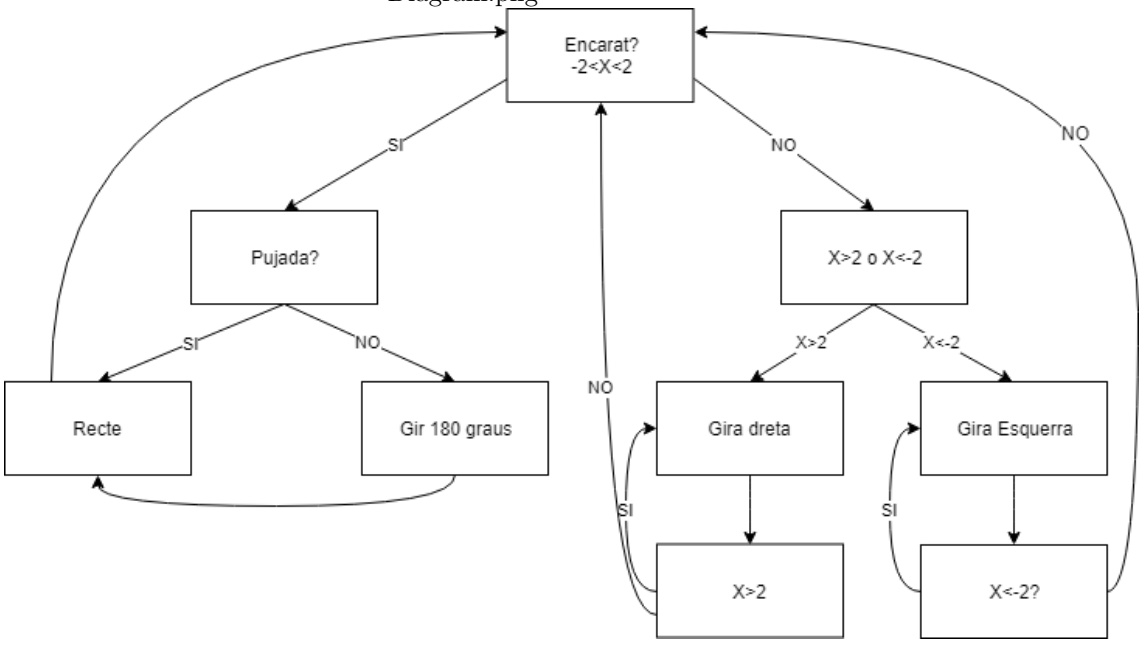
```
void magia(){

  float Lsp1, Rsp1;
  int opcio;
  float snsrY = (float)sensor.accelData.y * 9.8 / (float)(2 << 11);
  float snsrX = (float)sensor.accelData.x * 9.8 / (float)(2 << 11);

  if (snsrY >= 0){
    LSMandoTuneig(100,0,&Lsp1,&Rsp1);
    Motoreta.LsMtMando(Lsp1, Rsp1, 1);
    Motoreta.LsMtMotor();
    if (-2 < snsrX < 2){
      delay(1000);
     LSMandoTuneig(100,0,&Lsp1,&Rsp1);
     Motoreta.LsMtMando(Lsp1, Rsp1, 1);
     Motoreta.LsMtMotor();
    }
    else{
      while(-1 > snsrX || snsrX < 1){
       if (snsrX < 0){
         while (snsrX < -1){
           //Gira esquerra
           LSMandoTuneig(100,80,&Lsp1,&Rsp1);
           Motoreta.LsMtMando(Lsp1, Rsp1, 1);
           Motoreta.LsMtMotor();
           //Actualitza dades sensor
           sensor.readAccelData();
           snsrX = (float)sensor.accelData.x * 9.8 / (float)(2 << 11);
         }
       }
       else{
         while (snsrX > 1){
           //Gira dreta
           LSMandoTuneig(100,-80,&Lsp1,&Rsp1);
           Motoreta.LsMtMando(Lsp1, Rsp1, 1);
           Motoreta.LsMtMotor();
           //Actualitza dades sensor
           sensor.readAccelData();
           snsrX = (float)sensor.accelData.x * 9.8 / (float)(2 << 11);
         }
       }
       sensor.readAccelData();
           snsrX = (float)sensor.accelData.x * 9.8 / (float)(2 << 11);
      }
    }

  }
  else{

   if (-2 < snsrX < 2){
     //Gira 180 graus
    int snsrY2= (int) snsrY;

    while((int) snsrY != -snsrY2){
      LSMandoTuneig(100,80,&Lsp1,&Rsp1);
      Motoreta.LsMtMando(Lsp1, Rsp1, 1);
      Motoreta.LsMtMotor();
      sensor.readAccelData();
      snsrY = (float)sensor.accelData.y * 9.8 / (float)(2 << 11);
    }

   }
   else{
     if (snsrX < 0){
       while (snsrX < -1){
         //Gira esquerra
         LSMandoTuneig(100,80,&Lsp1,&Rsp1);
         Motoreta.LsMtMando(Lsp1, Rsp1, 1);
         Motoreta.LsMtMotor();
         //Actualitza dades sensor
         sensor.readAccelData();
         snsrX = (float)sensor.accelData.x * 9.8 / (float)(2 << 11);
       }
     }
     else{
       while (snsrX > 1){
         //Gira dreta
         LSMandoTuneig(100,-80,&Lsp1,&Rsp1);
         Motoreta.LsMtMando(Lsp1, Rsp1, 1);
         Motoreta.LsMtMotor();
         //Actualitza dades sensor
         sensor.readAccelData();
         snsrX = (float)sensor.accelData.x * 9.8 / (float)(2 << 11);
       }
     }
   }
  }
}
```

En un primer moment mirem el sensor Y de l'acceleròmetre per determinar si estem plans/inclinats cap a dalt, o cap a baix. En el cas de que estiguem plans o mirant cap a dalt, encara podria donar-se la situació de que estiguéssim molt desviats del punt més alt, pel que cal mirar els eixos de x. Els eixos de X, ens indiquen si una roda esta més elevada que la altra. Idealment haurien d'estar igualades, per tant a 0. A partir de fer proves hem decidit donar un marge de 2 positiu i negatiu, si està entre aquests dos casos anirà recte, i en el cas de que sigui inferior a -2 o superior a 2, rectificarà girant cap al costat que li toqui.
En el cas de que estigui mirant cap a baix, però que ambdues rodes estan equilibrades (entre -2 i 2) farem una volta de 180 graus. En els altres casos, girarem per mirar cap a dalt per la dreta o la esquerra, segons sigui més ràpid. Per exemple si la roda dreta està més elevada que la esquerra, serà més ràpid moure només la roda esquerra per equilibrar-la i que estigui encarat cap amunt el més ràpid possible.
En tots els casos, com es pot observar a la imatge del codi, fem que el robot vagi a la màxima velocitat, 100, per així intentar anar el més ràpid possible. Hem fet diferents provatures de provar de girar més lent per així mirar si era més exacte, però a la llarga sempre ens sortia que arribava al punt més alt de forma més ràpida si sempre anava a màxima velocitat.

Quan cal girar, dins del if que fa que giri cap a la part més elevada, tenim un bucle que va fent girar i fent una nova lectura dels sensors de l'acceleròmetre. D'aquesta manera, es queda dins del bucle fins que ha arribat dins dels marges que fan que el robot estigui encarat cap a la pujada.
També fem us de un while de "protecció", que s'encarrega de que si ens hem passat de girada, executi el gir contrari fins que estigui ben encarat.

2.3 Diagrama

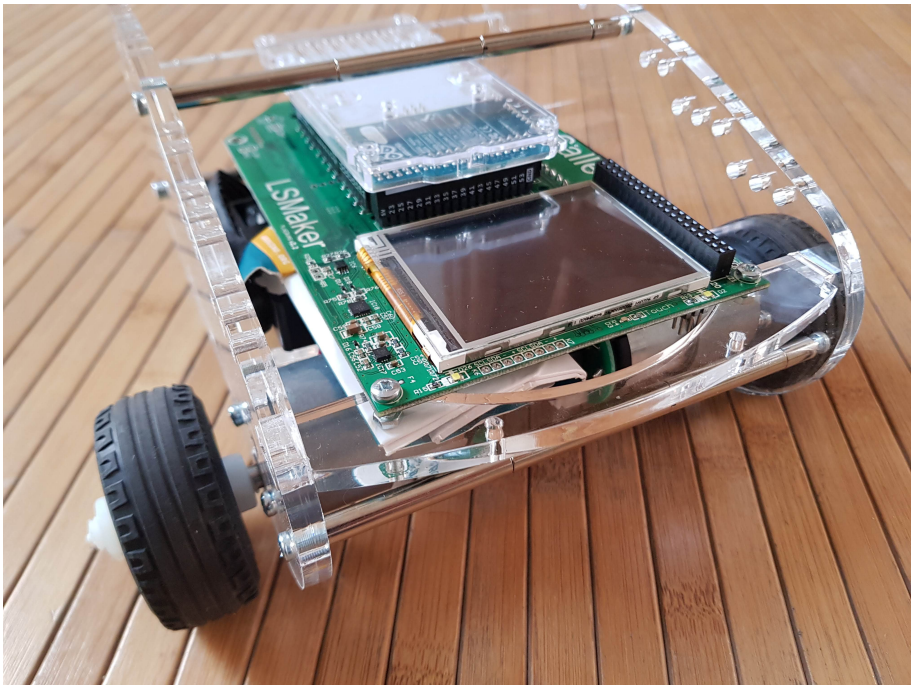
Diagram.png



3 Hardware

En el nostre cas hem fet servir el LsMaker V2, sense cap modificació per millorar el seu rendiment.
Si que ens hem trobat amb la necessitat de fer una modificació de hardware per el correcte funcionament del robot, D'aquesta ens vam adonar un cop ja fèiem les provatures finals. Estava realitzant una pujada de gran pendent quan de cop el robot va apagar-se. En un primer moment vam pensar que podia haver-se sobre-calentat, ja que no arrancava i feia una mica de olor.

Finalment no es tractava de un sobrecalentament, sinó que al ser un terreny una mica rugós les piles s'havien sortit una mica de lloc. Es per això que vam posar un parell de plàstics, un per limitar el moviment del aguantapiles horitzontal, i un altre per el vertical.



A la imatge podem veure el plàstic negre lateral, i el cartró cobert amb plàstic per sota que evita que les piles saltin a causa del terreny.

4 Conclusions

Aquest projecte ha sigut un dels més divertits de realitzar dels que hem fet fins ara a Universitat. També un dels més satisfactoris, ja que pots veure el resultat dels teus esforços de forma física i pots ensenyar-ho a amics i família i ho entendran fàcilment.
El fet de que es tracti de una competició també t'anima a anar un pas més enllà per intentar millorar al màxim el teu software i hardware. Un altre punt molt interessant es que estem treballant amb un codi de més baix nivell, que interactua directament amb el hardware. Això ens és molt útil per arribar a tenir un control total de que vols que faci el teu robot, tot hi que en certs moments pot arribar a ser una mica frustrant.

Ens ha obert molt els ull de quantes possibilitats tens amb el arduino. Ajuda molt que sigui un entorn de programació més familiar que no els dels PIC de compus, ja que es com programar en c i amb varies llibreries que t'ajuden a no tenir que estar tant pendent de saber quin port treu diferent informació...
Ens agradaria que haguessin més pràctiques amb el LsMaker, ja que aquest es el primer cop que realment el programem per quelcom una mica més complicat i interessant, i el resultat ens ha agradat molt.

5 Línies de futur

Durant la realització del projecte ens hem anat pensant en maneres de com poder fer que el nostre robot aconseguís el seu propòsit de forma més efectiva, però per falta de temps (i diners!) no ho hem dut a terme.
Una de les primeres idees que ens va sortir era la de modificar alguns dels condensadors i resistències dels motors del robot per veure si podíem fer que anés més ràpid, però degut a que som informàtics i no tenim gaire confiança en les nostres habilitats de electrònica vam deixar-ho córrer. Una altra opció hagués estat comprar-li uns motors nous més ràpids.

Un altre dels punts on hagués millorat és canviant la bola de ferro de la part del darrere del robot per un altre parell de rodes amb motors cada una, d'aquesta forma haguéssim, aconseguit tenir més velocitat i tracció, el qual ens hauria estat molt bé perquè degut a la bola de ferro a vegades ens derrapava del darrere.

En quant a software, creiem que potser podríem haver intentat buscar uns marges/valors més ajustats per a que intentés sempre anar cap al punt exacte, però vam trobar-nos bastant limitats per el acceleròmetre incorporat, ja que tot hi estar quiet a el terra els valors anaven fluctuant (poc) i això provoca que no poguéssim fer un programa molt exacte.

6 Bibliografia

Durant el projecte no hem fet cap tipus de recerca a internet, tot ha sigut mirar el projecte base proporcionat a estudy i anar investigant que feia cada funció. Prova i error.

SoftwareBase v4.4. [Accessed 19 December 2018]
Available from: https://estudy.salle.url.edu/course/view.php?id=3907

BAELDUNG. Quicksort Algorithm Implementation in Java. Baeldung [online]. 12 November 2018. [Accessed 19 December 2018].
Available from: https://www.baeldung.com/java-quicksort

Latex Expresiones Matematicas. Demostraciones Matematicas problemas ejercicios preguntas consultas dudas ayuda apoyo, tareas. Foros. Tex, Latex Editor. MathJax. Math help [online]. [Accessed 19 December 2018].
Available from: http://www.rinconmatematico.com/instructivolatex/formulas.htm

LaTeX Fácil. LaTeX Fácil: Guía rápida de LaTeX [online]. [Accessed 19 December 2018].
Available from: http://nokyotsu.com/latex/curso.html

Line breaks and blank spaces. Overleaf [online]. [Accessed 19 December 2018].
Available from: https://www.overleaf.com/learn/latex/Line$_b$reaks$_a$nd$_b$lank$_s$paces