

Pioneer 3DX: Entorn de Pràctiques

- Descripció Física del Robot
- Saphira/ARIA/AROS.
- Connexió al Robot/Simulador.
- Principals classes de ARIA: ArRobot.
- Compilar i Simular.
- Connexió al Robot Real

Robòtica

Pàg. 1

Pioneer 3DX: Descripció Física

- Robot de 3 rodes: 2 motrius i una 'lliure'
- Tracció diferencial.
- Posicionament mitjançant odometria:
 - Encoders



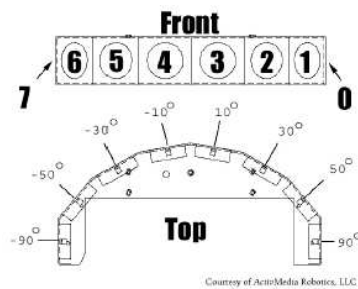
ENCODER TICKS/REV	500
GEAR RATIO	19.7
WHEEL DIAM (MM)	165
ENCODER TICKS/MM	76
DISTCONVFACTOR	0.840
DIFFCONVFACTOR	0.0056

Robòtica

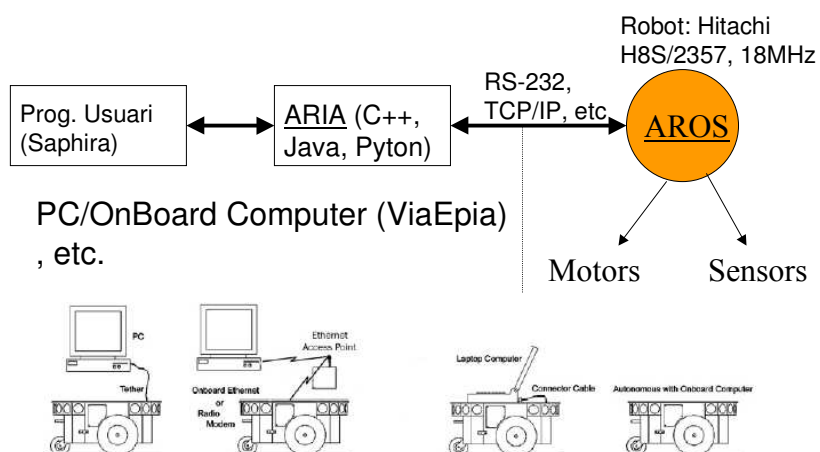
Pàg. 2

Ultrasons

- 2 arrays de 8 sensors d'ultrasons.
- Abast: 10cm- 4m.
- Freqüència de lectura: 25Hz (40 milisegons per sonar).



SAPHIRA/ARIA/AROS



ARIA

- ARIA: API per a accedir a les funcions de l'AROS des de qualsevol programa.
- Moure robot, lectura de sensors, etc.
- Exemple, programa de connexió al robot:

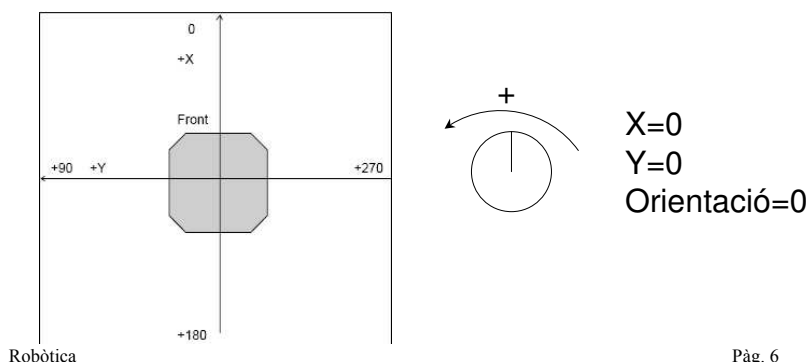
<pre>#include "Aria.h" int main(int argc, char **argv) { ArRobot robot3; Aria::init(); ArSimpleConnector connector(&argc, argv); if (!connector.parseArgs() argc > 1) { connector.logOptions(); exit(1); } if (!connector.connectRobot(&robot3)) { printf("Could not connect to robot.exiting\n"); Aria::shutdown(); return 1; } }</pre>	<pre>robot3.comInt(ArCommands::SOUNDTOG, 1); robot3.comInt(ArCommands::ENABLE, 1); robot3.runAsync(false); while(robot3.isRunning()) { //Aquí va el vostre codi } robot3.stopRunning(); Aria::shutdown(); return 0; }</pre>
---	--

Robòtica

Pàg. 5

Sistema de Coordenades

- Posició i orientació 0 en el moment de resetejar el robot.
- La resta de posicions absolutes es referencien a aquest sistema de coordenes inicial.
- Distàncies en mm. i angles en graus.



Robòtica

Pàg. 6

ArRobot (I)

- Classe que encapsula bona part de la funcionalitat del robot.
- Principals mètodes:
 - *move(double dist.)*: mou en línia recta la distància en mm.
 - *setHeading(double angle)*: orienta (gira) el robot fins al angle indicat.
 - *setAbsoluteMaxTransVel(mm/s)*: indica la vel. màxima de translació.
 - *setAbsoluteMaxRotVel (mm/s)*: indica la vel. màxima de rotació.
 - *isMoveDone(dist)*: retorna true si la darrera ordre de *move* ha finalitzat.
 - *isHeadingDone(angle)*: retorna true si la darrera ordre de *setHeading* ha finalitzat.

ArRobot (II)

- *setVel(mm/s)*: indica la velocitat de translació.
- *setRotVel()*: indica la velocitat de rotació.
- *getX()*, *getY()*: retorna la posició absoluta del robot en mm.
- *getTh()*: retorna la orientació absoluta del robot.
- *getNumSonar()*: retorna el nombre de sonars del robot.
- *getSonarRange(int i)*: retorna la lectura del ultrasó 'i'. Els ultrasó es comencen a comptar per 0.
- *isSonarNew(in i)*: indica si l'ultrasó 'i' té una nova lectura.

Exemple:

```
robot3.setAbsoluteMaxTransVel(300);
robot3.setHeading(45);
While(!robot3.isHeadingDone());
robot3.move(1000); //Moure 1 m.
While(!robot3.isMoveDone(10));

//El robot es mou 4s. En linea recta
robot3.setVel(200);
ArUtil::sleep(4*1000);
printf("x= %g, y= %g\n", robot3.getX(), robot3.getY());
```

ArRobot / ArSensorReading

- `getSonarReading (int i)`: retorna la lectura del sonar 'i' com un objecte de la classe **ArReading**.
- `ArSensorReading` encapsula la informació de la lectura d'un sensor (ultrasó). Mètodes:
 - `getX ()`, `getY()`: component X i Y de la lectura
 - `getSensorTh (void)`: retorna l'angle on està situat el sensor.
 - `getRange()`: retorna la distància de l'ostacle detectat.
- Altres Classes i mètodes:
 - `ArUtil::sleep(double milisegons)`.
 - `ArMath:atan2(x, y)`: `atan(y/x)`.

Tasques

- Es pot indicar al sistema que de manera periòdica (100 ms.) executi una determinada tasca.
- `bool ArRobot::addUserTask (const char * name, int position, ArFunctor * functor, ArTaskState::State * state = NULL)`
 - name: nom de la tasca
 - position: prioritat de la tasca, nombres majors indiquen prioritat major.
 - Functor: punter a la funció que s'ha d'executar.

Exemple d'una Tasca

```
class UpdatingTask
{
public:
    UpdatingTask(ArRobot *robot, int *sensorReadings);
    ~UpdatingTask(void);
    void doTask(void);
protected:
    ArRobot *myRobot;
    ArFunctorC<UpdatingTask> myTaskCB;
};

UpdatingTask::UpdatingTask(ArRobot *robot) :
myTaskCB(this, &UpdatingTask::doTask)
{
    myRobot = robot;
    //Afegir la tasca
    myRobot->addUserTask("UpdatingTask", 100,
                        &myTaskCB);
}
```

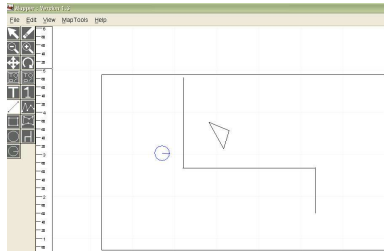
```
void UpdatingTask::doTask(void) {
    //Afegiu aquí el vostre codi
}



.....

void main(int argc, char **argv) {
    ArRobot robot;
    updatingTask ut(&robot);
    ....
}
```

Mapper

- Mapper: programa per a la creació de entorns (*.wld)



- En tot entorn heu de situar un sol robot 
- S'han de definir els límits de l'entorn com un rectangle amb 

Compilar i Simular

- Compilar: `compilar.sh`.
- Simulador a Linux: `/usr/local/Aria/bin/SRIsim`.
- `> /usr/local/Aria/bin/SRIsim`
- Càrrega de móns creats pel Mapper.
- Executar i connectar al simulador: `./exemple`
- Per defecte la connexió és al port 8101
- Per port sèrie:
 - `SRIsim -rp com2` `SRIsim -rp ttyS0`
 - Mes informació: `./SRIsim --help`

Implementació sobre robots reals

- Pioneer + ViaEpi connectada al port sèrie
- 510 MB Ram.
- Compact Flash 512 MB
- Wireless per a les comunicacions: 192.168.64.2
- Distribució Linux: Debian.
- Des de un ordinador carregar els vostres programes fonts al robot: sftp 192.168.64.2
- Obriu sessió: ssh 192.168.64.2
- Compilar i executar els programes al robot

Implementació sobre robots reals (II)

- No atureu el robot sense haver aturat correctament l'ordinador
shutdown -h now&& exit
- Important: no feu les proves sobre el robot real a una velocitat superior a 100.
- Més informació:
 - /usr/local/Aria/docs
 - <http://robots.activmedia.com/>