Πανεπιστήμιο Θεσσαλίας, 2011-2012

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Θέμα:

«Μελέτη κινητικότητας σε ασύρματα δίκτυα με τη χρήση ρομπότ και πειραματικών διατάξεων»

Χαρίλαος Νιαβής

<u>Επιβλέποντες καθηγητές:</u>

Λέανδρος Τασιούλας

(Καθηγητής)

<u>Συνεπιβλέπων καθηγητής:</u>

Ιορδάνης Κουτσόπουλος

(Επίκουρος Καθηγητής)

<u>2^{ος} Συνεπιβλέπων καθηγητής:</u>

Αθανάσιος Κοράκης

Βόλος Σεπτέμβριος 2011

Ευχαριστίες

Ύστερα από μια πορεία πέντε και πλέον ετών στο Τμήμα Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, ολοκληρώνω τις προπτυχιακές μου σπουδές με την εκπόνηση της παρούσας διπλωματικής εργασίας.

Θα ήθελα αρχικά να ευχαριστήσω θερμά τον κ. Κοράκη Αθανάσιο, του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, για τις χρήσιμες συμβουλές και υποδείξεις του καθώς και για την υποστήριξη που μου πρόσφερε κατά τη διάρκεια της φοίτησής μου, αλλά και κατά την εκπόνηση της διπλωματικής μου εργασίας.

Ευχαριστώ επίσης τον επιβλέποντα της εργασίας μου, Καθηγητή του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, κ. Λέανδρο Τασιούλα και τον συνεπιβλέποντα Επίκουρο Καθηγητή του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, κ. Ιορδάνη Κουτσόπουλο για την καθοδήγησή τους.

Από καρδιάς θα ήθελα να ευχαριστήσω τον κ. Νίκο Γιαλλελή, υποψήφιο Διδάκτορα του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων και τον κ. Γιάννη Καζδαρίδη, μεταπτυχιακό φοιτητή του Τμήματος Μηχανικών Η/Υ Τηλεπικοινωνιών και Δικτύων, για την πολύτιμη συνδρομή τους στο να φέρω σε πέρας την διπλωματική μου εργασία.

Τέλος, ευχαριστώ θερμά την οικογένειά μου για την αμέριστη συμπαράσταση που μου παρείχε όλα αυτά τα χρόνια για την ολοκλήρωση των προπτυχιακών σπουδών μου.

Στην οικογένειά μου

CONTENTS

ABSTRACT

The present Final Project Dissertation - Thesis is an implementation of a mobile node of NitLab testbed.

The first part of the project, consists of an introduction about wireless networks and mobility in wireless networks.

At the second part, we briefly present iRobot Create, which is the robot we used for our mobile node. We concentrate at the iRobot Create Open Interface, which gives us the capability of controlling Create’s behavior and reading its sensors. We present also the board we used for our robot.

At the third part lies the implementation of the robot. Firstly an analysis of the parts of our robot gives an idea of what it is constructed. Finally we give the code we used for navigation and for image recognition.

1. Introduction

Network simulation continues to be the dominant method of experimental evaluation in wireless networking. However, much research has established the failure of simulator models to adequately express wireless signal propagation. These shortcomings can lead to incomplete evaluation of wireless protocols and applications. When wireless research includes mobility, real evaluation becomes still more difficult due to the difficulty of creating and controlling mobile nodes in a real environment.

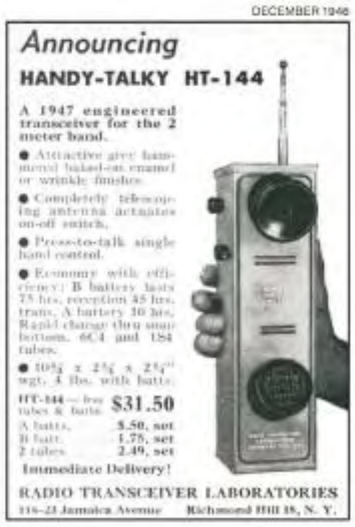
1.1 Wireless Networks in a Nutshell

The radio frequency spectrum is divided into several bands, starting as low as 30 kHz for maritime communication and ranging up to 300 GHz. Low frequency radio waves easily pass through human bodies, walls, water, etc. Higher frequencies are absorbed and reflected by obstacles. There are numerous other facts to be told from physics. For some frequency bands the ionosphere reflects signals. The background noise level differs in various bands. Signal strength is influenced by obstruction, diffusion, reflection, multi path fading, and the Doppler effect. A man-made problem is the interference of radio signals of multiple senders. Furthermore, in theory the signal strength is fading with a power of two with respect to the distance, which is only true in empty space. In other environments the exponent is larger and can have values from 3-5. This implies that if one tries to send over a distance of d transmission power has to chosen ~ d^2 in empty space. Many transmission models assume that the covered area by a radio signal can be modeled by a disc. However, practical measurements show that this is not at all the case.

1.2 Mobility in Wireless Networks

Today, it is hard to imagine the difficulties to send information over large distances before the invention of radio communication. First, such devices were so large and heavy that they could not be carried around but had to be carried by vehicles. So, the history of mobile communication starts with radio devices on boats which emerged in the

1890s and helped ocean vessels to overcome insulation for navigation and emergency situations. It took some time until mobile radio transceiver could be used on non-marine vehicles. Such technology was available in the 1930s when a radio transceiver could be operated on a bicycle and a radio sender could be operated on an airplane. Both as show cases and not really for practical use. At the end of the 30s portable solutions were available in form of the famous “walky-talky”, which could be carried by a single person. In the beginning of the 1940s a radio transceiver was available which could be held in a single hand: “The Handy-Talky”, see Fig. 1. Needless to say that these communication devices played an important role in the second world war. With the upcoming of transistors, large scale integrated chip layout, and new battery technology allowed the size of radio devices to shrink unbelievably small. Today, one can buy fully equipped sensor nodes with radio transceiver and micro-controller in the size of a small coin (and within reasonable price), see Fig. 2. Furthermore, there is the vision of communication devices being so small called “smart dust”.



DECEMBER 1946

Announcing

HANDY-TALKY HT-144

A 1947 engineered transceiver for the 2 meter band.

- Attractive gray hammered baked-on enamel or wrinkle finishes.
- Completely telescoping antenna actuates on-off switch.
- Press-to-talk single hand control.
- Economy with efficiency! B battery lasts 75 hrs. reception 45 hrs. trans. A battery 10 hrs. Rapid change thru snap bottom. 6C4 and 1S4 tubes.
- 10¾ x 2⅝ x 2¾" wgt. 4 lbs. with batts.

HT-144 — less tubes & batts.	**$31.50**
A batts.	5.50, set
B batt.	1.75, set
2 tubes	2.49, set

Immediate Delivery!

RADIO TRANSCEIVER LABORATORIES
116-23 Jamaica Avenue   Richmond Hill 18, N. Y.

Fig. 1. The Handy-Talky

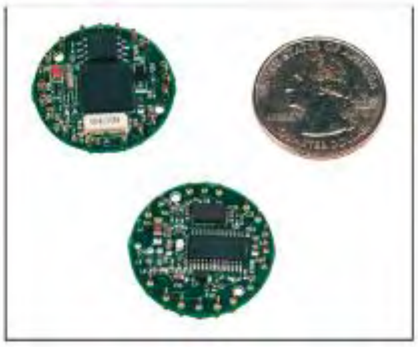


Fig. 2. The Mica2Dot from Crossbow

Also the underlying radio communication technology has changed much from the analogous broadcast signal (still surprisingly widely distributed) to digital multi-hop networks with time, frequency and code division multiplexing for the parallel use of the medium. Such packet oriented radio devices have been developed in the 1970s and filled the interior of a van, i.e. Vint Cerf’s Stanford Research Institute (SRI) van. Then, the packet radio underwent a miniaturization process and packet radio has become the dominating radio technology, so far.

For two-way radio communication central radio stations are used which serve as relay station for transmitting the radio signals. Many of these central relay stations partition the radio landscape into cells. In such cellular networks the mobility of users (more or less) reduces to problems of handover from one radio station to a neighbored station. Networks without such centralized infrastructure are called mobile ad hoc networks. There, the impact of mobility is much higher, since everything is moving. One can easily imagine the possible negative impact of mobility on wireless networking. Recent results point out that mobility has also a positive impact.

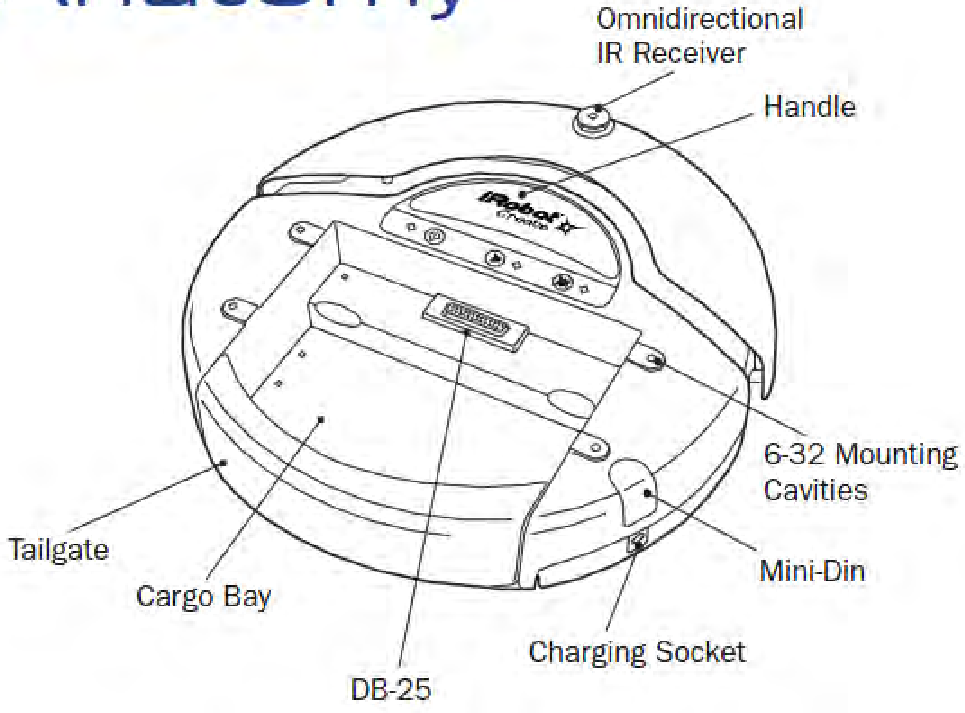
1.3 Mobility is helpful

One might think that mobility has only a negative impact on the behavior of wireless networks. But recent work has shown that this is not the case. Mobility improves the coverage of wireless sensor networks. It helps security in ad hoc networks. Furthermore it can help with network congestion as shown in [3]. This approach overcomes the natural lower bound for throughput of $\Omega(\sqrt{n})$ by instrumenting the random movement of nodes. They design a protocol where mobile nodes relay packets and literally transport them towards the destination node.

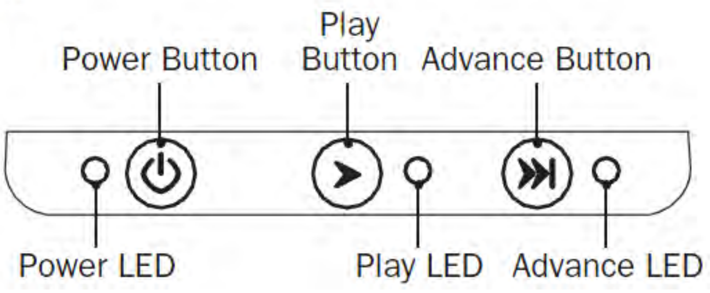
2. About IRobot Create

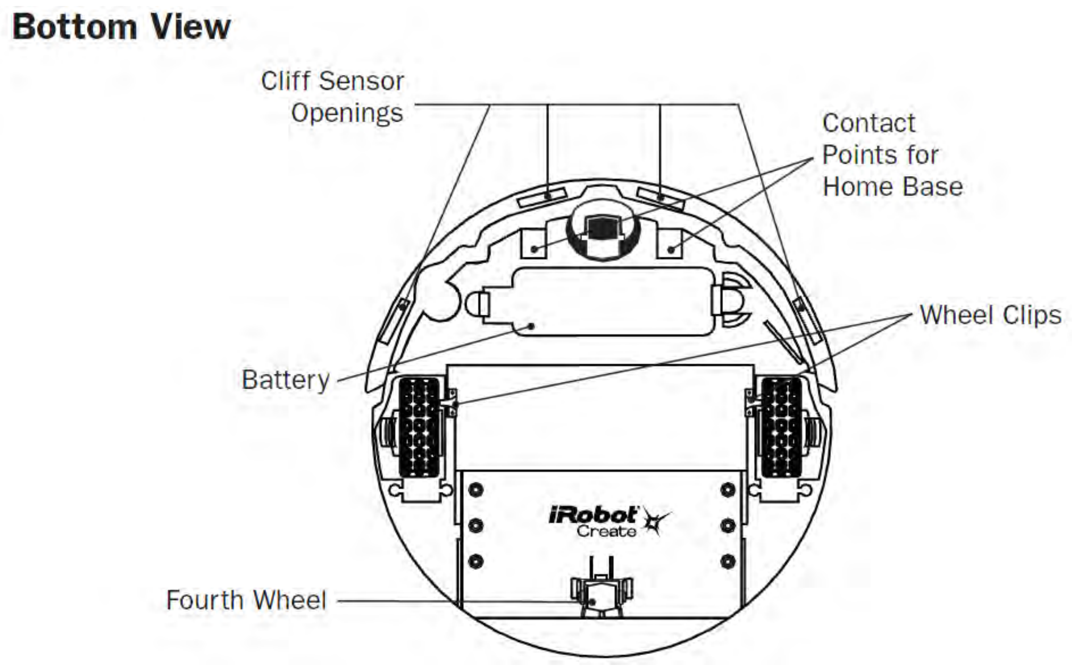
iRobot® Create is a complete robot development kit that allows you to program new robot behaviors without having to worry about mechanical assembly and low-level code. iRobot Create's Open Interface (OI) provides you with a set of commands, such as "drive" commands, demo commands, song commands and sensor commands, that encourage your own inventiveness. With iRobot Create, you can develop new behaviors and add third party electronics, all without having to worry about mechanical robustness or low-level control. When you first begin working with iRobot Create you may find its ten built in demos useful as inspirations for your own ideas. As you become more comfortable with its abilities and interface you will want to write your own Open Interface-based programs to control iRobot Create using a serial connection from a PC or microprocessor such as the iRobot Command Module. You can attach and control other hardware and electronic devices to iRobot Create; such as a robotic arm, light display, or a ranging sensor. It can also become a musical instrument! By using the Song command you can write and save up to 16 songs in iRobot Create.

Anatomy



Buttons and Lights





2.1 Demos

iRobot Create comes with ten built-in demos to provide inspiration for your own robot programming adventures.

• Press the power button and wait for the power LED to stop flashing.
• To select a demo, press the Advance button . iRobot Create beeps the number of the selected demo. One long, low beep is equal to five short, high beeps. For example, when you select demo number 8, iRobot Create plays long-short-short-short (_...).
• iRobot Create's Advance LED blinks in the same pattern while a demo is running or paused.
• Press the Play button to start the currently selected demo.
• To stop the demo, press the Play button or the Advance button.

Demo Descriptions:

Demo	Description	Demo #	Demo Beep Pattern
Cover	iRobot Create covers an entire room using a combination of behaviors, such as bouncing off of walls, following walls, and spiraling.	1	·
Cover and Dock	Identical to the Cover demo, with one exception; if iRobot Create sees the Home Base's* infrared signals, it uses these to move towards the Home Base and dock with it.	2	··
Spot Cover	iRobot Create spirals outward, then inward, to cover an area around its starting position.	3	···
Mouse	iRobot Create tries to follow around the edges of a room using its wall sensor and bumper.	4	····
Figure Eight	iRobot Create continuously drives in a figure 8 pattern.	5	—
Wimp	iRobot Create drives forward when pushed from behind. If iRobot Create hits an obstacle while driving, it drives away from the obstacle.	6	—·
Home	iRobot Create drives toward a Virtual Wall* when the back and sides of its Omnidirectional IR Receiver are covered with black electrical tape. When it touches the Virtual Wall or another obstacle, it stops.	7	—··
Tag	Identical to the Home demo, except iRobot Create goes back and forth between multiple Virtual Walls by bumping into one, turning around, driving to the next Virtual Wall, bumping into it and turning around to bump into the next Virtual Wall.	8	—···
Pachelbel	iRobot Create plays the notes of Pachelbel's Canon in sequence when its cliff sensors are activated.	9	—····
Banjo	iRobot Create's four cliff sensors play the notes of a chord, depending on how the bumper is pressed: •No bumper: G major •Right or left bumper: D major7 •Both bumpers (center): C major	10	— —

2.2 Powering iRobot Create

iRobot Create can be powered with the included alkaline battery pack, which takes "AA" alkaline batteries, or with any of the iRobot rechargeable batteries. iRobot Create's power light indicates the charge state of the batteries. It is green when the batteries are fully charged or new, and gradually turns red as the batteries discharge.

Color of Power Light	Battery Status
Slow Pulsing Orange	Charging (iRobot rechargeable battery only)
Fast Pulsing Orange	Reconditioning Charge (iRobot rechargeable battery only)
Green	Fully Charged
Amber	Partially Discharged
Red	Almost Fully Discharged
Flashing Red	Fully Discharged

Alkaline Battery Pack

The included alkaline battery pack requires 12 non-rechargeable “AA” batteries to power iRobot Create. iRobot Create’s low-voltage shutoff is designed for alkaline batteries only; rechargeable batteries can be deeply discharged and damaged if they are used in the alkaline battery pack. iRobot Create does not attempt to charge batteries in the alkaline battery pack. Alkaline batteries will power the robot (on a hard floor with no payloads or attachments) for up to 1.5 hours if the robot is constantly moving. If you plan to have your robot moving continuously.

Using the Alkaline Battery Pack

Use a Phillips head screwdriver to loosen the four screws on the underside
of the green Alkaline Battery Pack. The screws will not come all the way out
of the doors. Slide the doors up and out of the batter y case. Inser t six
non-rechargeable alkaline “AA” batteries into each side of the battery case
and then slide the doors back on. Tighten the screws.
Insert the battery case into iRobot Create’s Battery Well, pushing until it clicks into place.

Self Charging Home Base

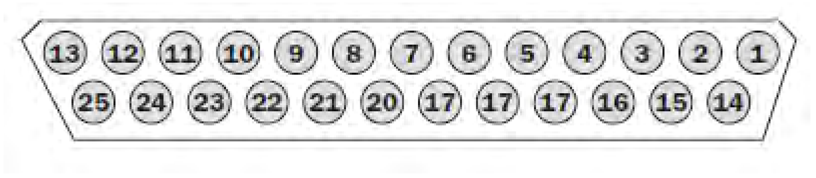
The Self Charging Home Base has three infrared transmitters that you can use in your applications. The Self Charging Home Base also enables iRobot Create to automatically charge its iRobot rechargeable battery. During two demos, iRobot Create will automatically return to the home base. During the Cover demo, if the battery is running low and the Omnidirectional IR Receiver on iRobot Create sees the Home Base IR transmitter, iRobot Create will dock itself on the Home Base for recharging. During the Cover and Dock demo, as soon as the Omnidirectional IR Receiver on Create see the Home Base IR transmitter, iRobot Create will dock itself on the Home Base for recharging.
While recharging, iRobot Create’s Power LED and the Home Base Dock lights will be green. The Home Base must be on a hard level surface and

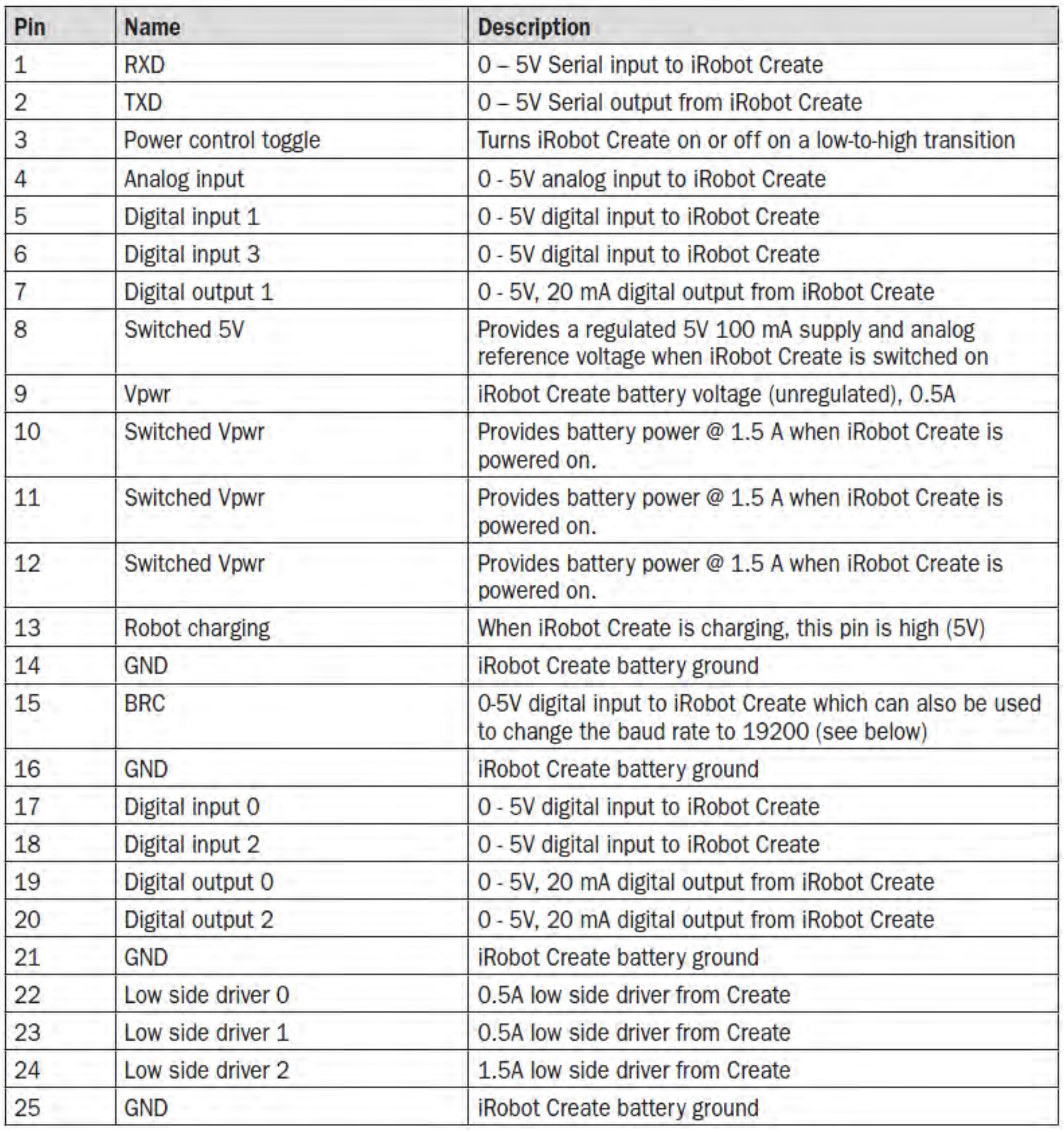
in an area where iRobot Create has a clear path to approach the Base. If iRobot Create is unable to dock on its first attempt, it will continue to try until it is successful

2.3 Cargo Bay Connector

The cargo bay connector, located in the front middle of the cargo bay contains 25 labeled pins that you can use to attach electronics and other peripheral devices such as additional sensors, lights, or motors to iRobot Create. The cargo bay connector provides four digital inputs, an analog input, three digital outputs, three low-side driver outputs (useful for driving motors), a charging indicator, a power toggle, serial Tx and Rx, a 5V reference, battery ground and battery voltage.

Cargo Bay Connector Pin Descriptions:





Pin	Name	Description
1	RXD	0 – 5V Serial input to iRobot Create
2	TXD	0 – 5V Serial output from iRobot Create
3	Power control toggle	Turns iRobot Create on or off on a low-to-high transition
4	Analog input	0 - 5V analog input to iRobot Create
5	Digital input 1	0 - 5V digital input to iRobot Create
6	Digital input 3	0 - 5V digital input to iRobot Create
7	Digital output 1	0 - 5V, 20 mA digital output from iRobot Create
8	Switched 5V	Provides a regulated 5V 100 mA supply and analog reference voltage when iRobot Create is switched on
9	Vpwr	iRobot Create battery voltage (unregulated), 0.5A
10	Switched Vpwr	Provides battery power @ 1.5 A when iRobot Create is powered on.
11	Switched Vpwr	Provides battery power @ 1.5 A when iRobot Create is powered on.
12	Switched Vpwr	Provides battery power @ 1.5 A when iRobot Create is powered on.
13	Robot charging	When iRobot Create is charging, this pin is high (5V)
14	GND	iRobot Create battery ground
15	BRC	0-5V digital input to iRobot Create which can also be used to change the baud rate to 19200 (see below)
16	GND	iRobot Create battery ground
17	Digital input 0	0 - 5V digital input to iRobot Create
18	Digital input 2	0 - 5V digital input to iRobot Create
19	Digital output 0	0 - 5V, 20 mA digital output from iRobot Create
20	Digital output 2	0 - 5V, 20 mA digital output from iRobot Create
21	GND	iRobot Create battery ground
22	Low side driver 0	0.5A low side driver from Create
23	Low side driver 1	0.5A low side driver from Create
24	Low side driver 2	1.5A low side driver from Create
25	GND	iRobot Create battery ground

2.4 Serial Port Settings

Baud: 57600 or 19200 (see below)
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None.

By default, iRobot Create communicates at 57600 baud. If you are using a microcontroller that does not support 57600 baud, there are two ways to force Create to switch to 19200:

Method 1:
When powering on Create, hold down the Play button. After about 4 seconds, Create plays a tune of descending pitches. Create will communicate at 19200 baud until the power is turned off, the battery is removed and reinserted, the battery voltage falls below the minimum required for processor operation, or the baud rate is explicitly changed by way of the OI.
Method 2:
Use the Baud Rate Change pin (pin 15 on the Cargo Bay Connector/pin 5 on the Mini-DIN connector) to change Create' s baud rate. After turning on Create, wait 2 seconds and then pulse the Baud Rate Change low three times. Each pulse should last between 50 and 500 milliseconds. Create will communicate at 19200 baud until the processor loses battery power or the baud rate is explicitly changed by way of the OI.

2.5 Irobot Create Open Interface

2.5.1 Overview

The Create Open Interface (OI) consists of an electronic interface and a software interface for controlling Create's behavior and reading its sensors. The electronic interface includes a 7 pin Mini-DIN connector and a DB-25 connector in the Cargo Bay for connecting hardware and electronics for sensors and actuators such as a robotic arm or light sensor to Create. The software interface lets you manipulate Create's behavior and read its sensors through a series of commands that you send to Create's serial port by way of a PC or microcontroller that is connected to the Mini-DIN connector or Cargo Bay Connector.

Open Interface Commands

The types of commands available include:
• Demo commands
• Driving commands
• Song commands
• Sensor commands
• Cargo Bay Connector commands
• Scripting commands

You send these commands to iRobot Create using the included serial cable or any other serial connection to a PC. The serial cable plugs into iRobot Create's Mini-Din connector, which is located in the rear right side of iRobot Create, beneath a snap-away plastic guard.

Sending OI Commands from a PC

Connect the serial cable between iRobot Create and your PC's serial port and start a serial terminal program that is capable of sending data. Configure the serial settings to 57600 baud, 8 data bits, 1 stop bit, and no flow control. Select the "Send" tab on RealTerm. Enter commands and watch the robot perform!

Examples:
• To turn on iRobot Create's Play LED only:
128 132 139 2 0 0

• To read the state of iRobot Create's left cliff sensor:
128 142 9
See how the robot's response changes when you lift it up.

• To make iRobot Create sing:
Send these command sets separately:
128 132 (Puts the robot in Full mode)
140 0 4 62 12 66 12 69 12 74 36 (Defines the song)
141 0 (Plays the song)

• To make iRobot Create drive forward:
CAUTION - place iRobot Create on the ground and be ready to grab it when it reaches the end of its serial cable tether! Send these two command sets separately:
128 131
137 0 100 128 0

You can also write scripts of commands, freeing your robot from the serial cable tether. A script is a sequence of commands including commands that wait for events. With a script you can make the robot perform a "song-anddance" routine that you create.

To make the robot perform more complex behaviors, you can write software that runs on the PC and sends commands to the robot while reading its sensor data. Your software can communicate with iRobot Create over the included serial cable or a wireless serial accessory.

Sending OI Commands from a Microprocessor

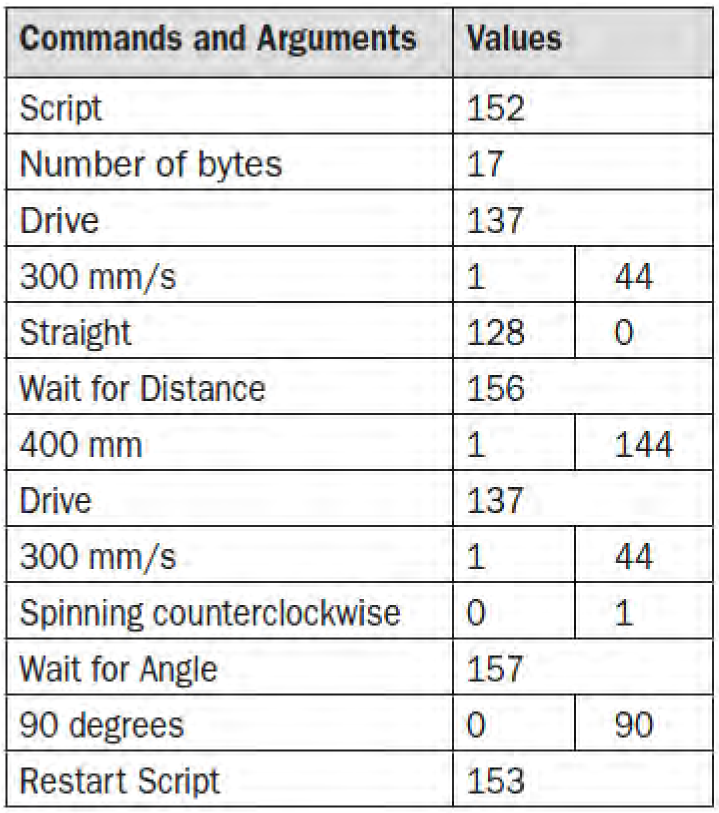
With a microprocessor like iRobot's Command Module attached to your robot, you can make it perform complex behaviors without a serial tether or wireless connection to a PC. With the iRobot Command Module, which plugs into iRobot Create's Cargo Bay Connector, you can write your own programs in C or C++, that send OI commands to iRobot Create and read its sensor data. You can also expand on the sensors and actuators that iRobot Create offers by adding your own custom hardware to the iRobot Command Module's four DB-9 expansion ports

2.5.2 Using the Script Command

To load a script onto iRobot Create, use the Script command (152), followed by the number of bytes in the script, followed by the series of commands and their arguments. Scripts can be up to 100 bytes long.

Example:
Drive in a square: 152 17 137 1 44 128 0 156 1 144 137 1 44 0 1 157 0 90 15

The meaning of the above string of numbers is:



Commands and Arguments	Values	
Script	152	
Number of bytes	17	
Drive	137	
300 mm/s	1	44
Straight	128	0
Wait for Distance	156	
400 mm	1	144
Drive	137	
300 mm/s	1	44
Spinning counterclockwise	0	1
Wait for Angle	157	
90 degrees	0	90
Restart Script	153	

Using the Play Script Command

When you are ready for iRobot Create to perform the script, send the Play Script command (153). This loads the script into the serial queue. iRobot Create then immediately begins executing the script's commands as if they had been sent directly to the serial port.

2.5.3 iRobot Create Open Interface Modes

The Create OI has four operating modes: Off, Passive, Safe, and Full. After a battery change or when is first supplied, the OI is in "off" mode. When it is off, the OI listens at the default baud rate (57600 or 19200 - see Serial Port Settings above) for an OI Start command. Once it receives the Start command, you can enter into any one of the four operating modes by sending a mode command to the OI. You can also switch between operating modes at any time by sending a command to the OI for the operating mode that you want to use.

Passive Mode

Upon sending the Start command or any one of the demo commands (which also starts the specific demo, e.g., Spot Cover, Cover, Cover and Dock, or Demo), the OI enters into Passive mode. When the OI is in Passive mode, you can request and receive sensor data using any of the sensors commands, but you cannot change the current command parameters for the actuators (motors, speaker, lights, low side drivers, digital outputs) to something else. To change how one of the actuators operates, you must switch from Passive mode to Full mode or Safe mode. While in Passive mode, you can read Create’ s sensors, watch Create perform any one of its ten built-in demos, and charge the battery.

Safe Mode

When you send a Safe command to the OI, Create enters into Safe mode. Safe mode gives you full control of Create, with the exception of the following safety-related conditions:

• Detection of a cliff while moving forward (or moving backward with a small turning radius, less than one robot radius).

• Detection of a wheel drop (on any wheel).

• Charger plugged in and powered.

Should one of the above safety-related conditions occur while the OI is in Safe mode, Create stops all motors and reverts to the Passive mode.

If no commands are sent to the OI when in Safe mode, Create waits with all motors and LEDs off and does not respond to Play or Advance button presses or other sensor input.

Note that charging terminates when you enter Safe Mode.

Full Mode

When you send a Full command to the OI, Create enters into Full mode. Full mode gives you complete control over Create, all of its actuators, and all of the safety-related conditions that are restricted when the OI is in Safe mode, as Full mode shuts off the cliff, wheel-drop and internal charger safety features. To put the OI back into Safe mode, you must send the Safe command.

If no commands are sent to the OI when in Full mode, Create waits with all motors and LEDs off and does not respond to Play or Advance button presses or other sensor input.

Note that charging terminates when you enter Full Mode.

3. About ALIX Boards

Designed by PC Engines in Switzerland, the Alix single board computer is highly power efficient, small, and capable of running operating systems like Linux and FreeBSD unmodified thanks to its x86 compatible processor. These little boards can be found in small offices, datacenters and in the homes of computer enthusiasts, doing just about anything an embedded computer can do; routers, firewalls, mail servers, and network attached storage are all common roles for the Alix.

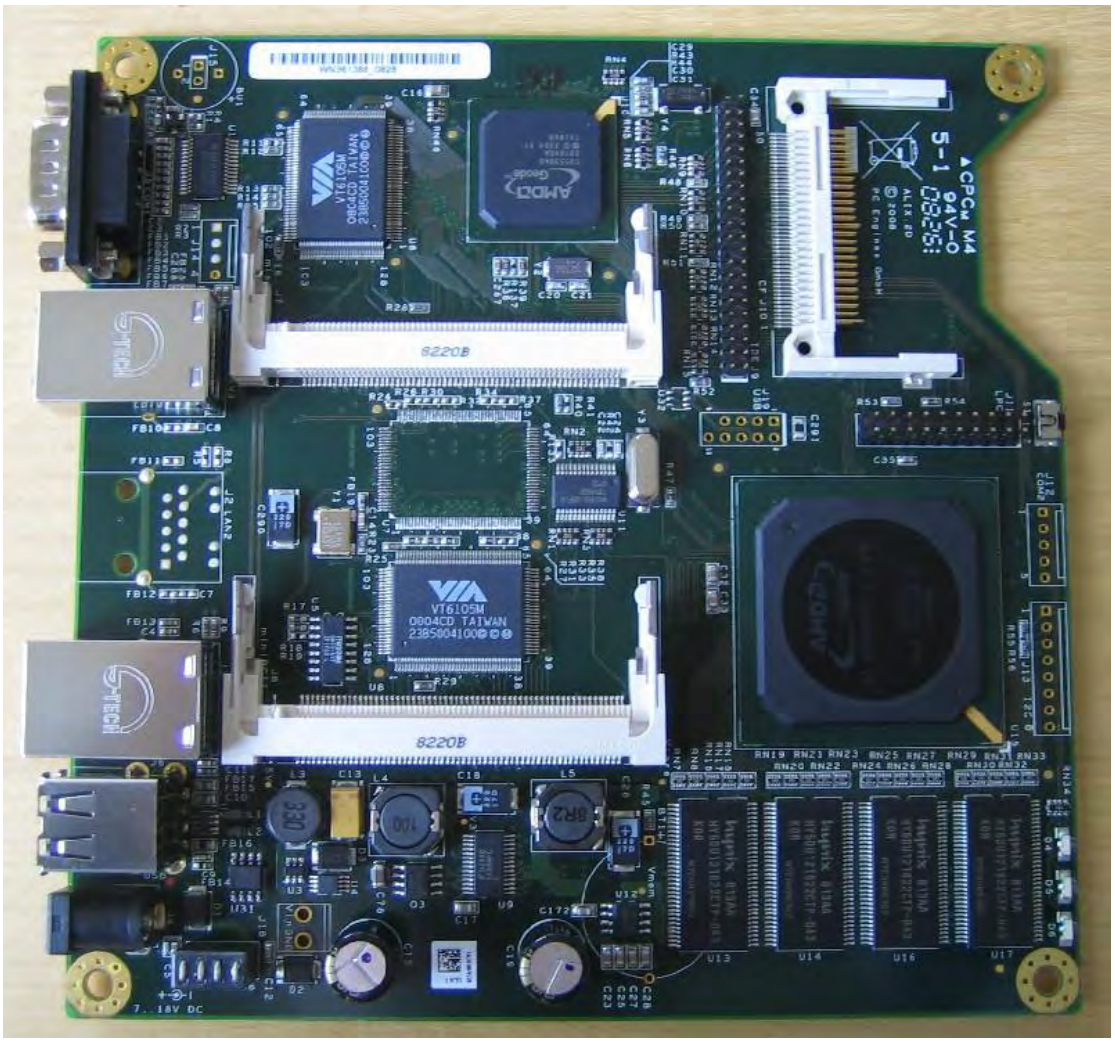
Our Alix Board

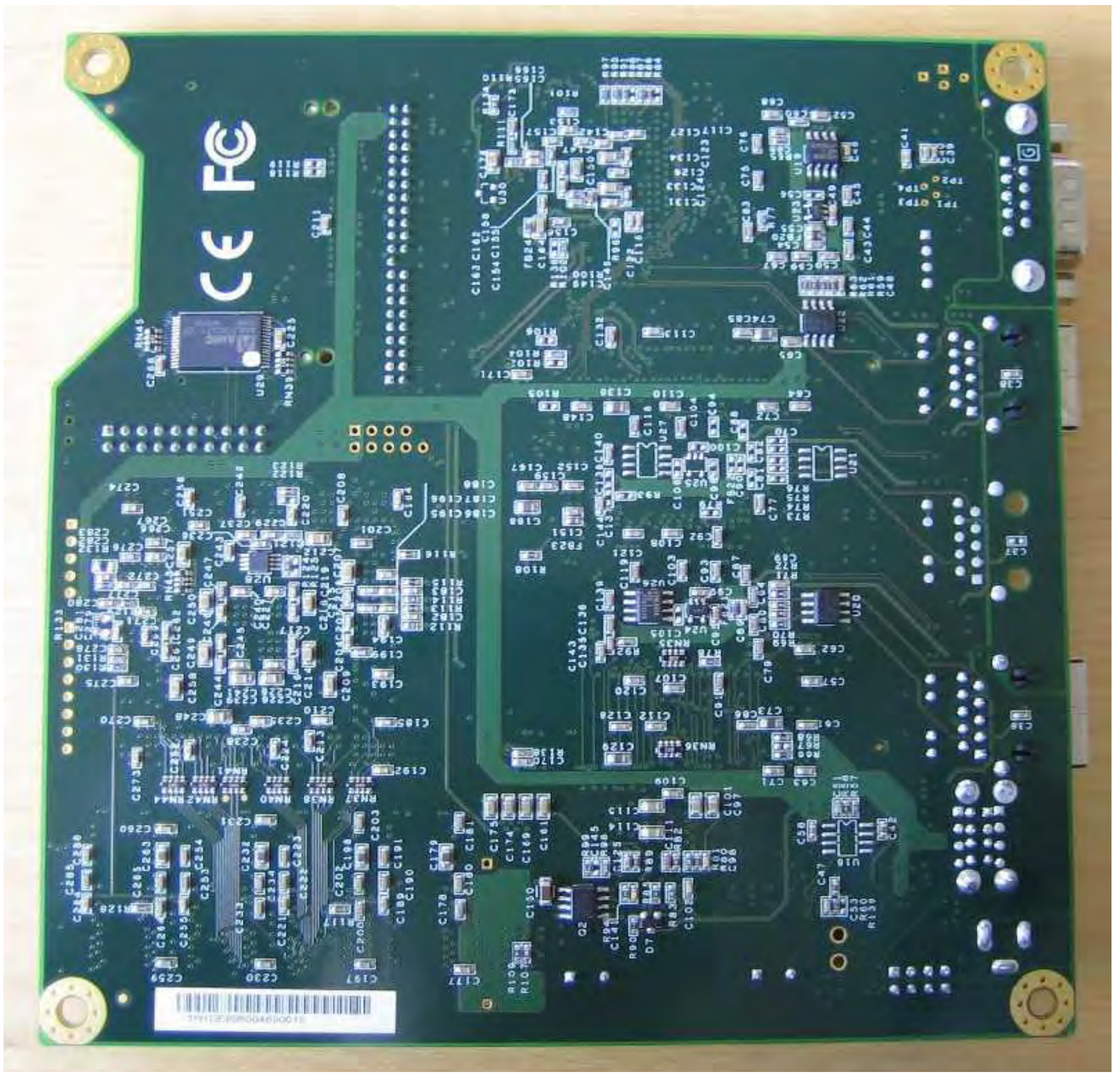
In our implementation we use alix2d2 system board.

Specifications:

- CPU: 500 MHz AMD Geode LX800
- DRAM: 256 MB DDR DRAM
- Storage: CompactFlash socket, 44 pin IDE header
- Power: DC jack or passive POE, min. 7V to max. 20V
- Three front panel LEDs, pushbutton
- Expansion: 2 miniPCI slots, LPC bus
- Connectivity: 2 Ethernet channels (Via VT6105M 10/100)
- I/O: DB9 serial port, dual USB port
- Board size: 6 x 6" (152.4 x 152.4 mm) - same as WRAP.1E
- Firmware: tinyBIOS

There is no VGA, keyboard or mouse connection, as this is strictly an embedded/headless device, however other models of the Alix do include those features.





4. Our Robot/Parts

It's an iRobot Create with a costum designed tray to hold the Alix Board. The board can send commands to the robot over a serial port and is powered off the robot's own battery. The Alix board runs Voyage Linux and therefore can be part of an experiment in the future. The upward facing camera is used to read specific patterns on the ceiling, that the robot uses to navigate around the room.

4.1 Storage

We use a 1gb compact flash memory card as a hard disk.

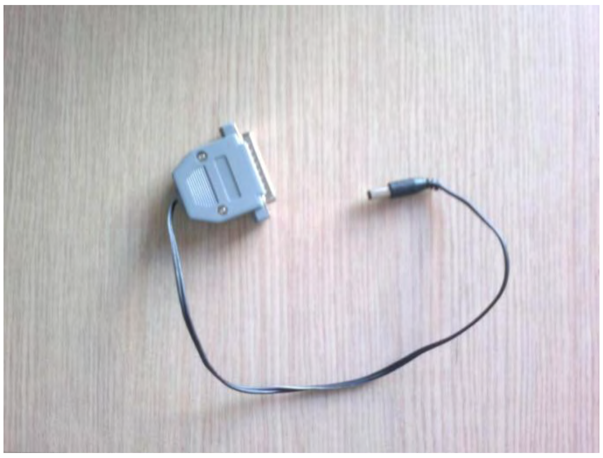
4.2 Power Supply

The Alix Board requires a min 7V to max 20V DC power. We had to power the board using Irobot Create. So we made a power connector for it.

Irobot Create comes with a transformer from 120VAC to 22VDC. We made a transformer from 220VAC to 110VDC to achieve charging.

4.3 Power Connector

As mentioned above, we are powering the board using the Create's battery. Fortunately, iRobot gives us access to the Create's battery through the 25-pin cargo bay connector.

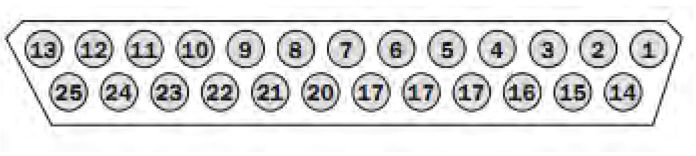


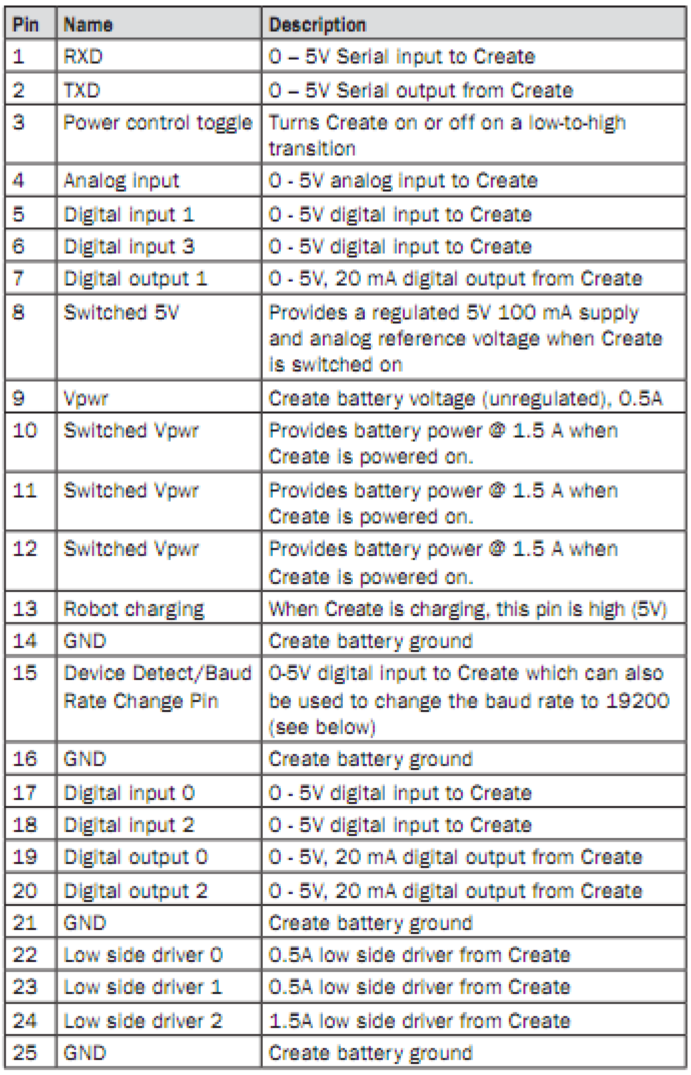
Connector assembly:

A male D-sub Connector is used to connect to the cargo bay connector:

- *25-Position Male Solder D-sub Connector*

- *25-Position Non-Shielded D-sub Connector Hood*





Pin	Name	Description
1	RXD	0 – 5V Serial input to Create
2	TXD	0 – 5V Serial output from Create
3	Power control toggle	Turns Create on or off on a low-to-high transition
4	Analog input	0 - 5V analog input to Create
5	Digital input 1	0 - 5V digital input to Create
6	Digital input 3	0 - 5V digital input to Create
7	Digital output 1	0 - 5V, 20 mA digital output from Create
8	Switched 5V	Provides a regulated 5V 100 mA supply and analog reference voltage when Create is switched on
9	Vpwr	Create battery voltage (unregulated), 0.5A
10	Switched Vpwr	Provides battery power @ 1.5 A when Create is powered on.
11	Switched Vpwr	Provides battery power @ 1.5 A when Create is powered on.
12	Switched Vpwr	Provides battery power @ 1.5 A when Create is powered on.
13	Robot charging	When Create is charging, this pin is high (5V)
14	GND	Create battery ground
15	Device Detect/Baud Rate Change Pin	0-5V digital input to Create which can also be used to change the baud rate to 19200 (see below)
16	GND	Create battery ground
17	Digital input 0	0 - 5V digital input to Create
18	Digital input 2	0 - 5V digital input to Create
19	Digital output 0	0 - 5V, 20 mA digital output from Create
20	Digital output 2	0 - 5V, 20 mA digital output from Create
21	GND	Create battery ground
22	Low side driver 0	0.5A low side driver from Create
23	Low side driver 1	0.5A low side driver from Create
24	Low side driver 2	1.5A low side driver from Create
25	GND	Create battery ground

- *25-Position D-sub Connector pins*

Power cord assembly:

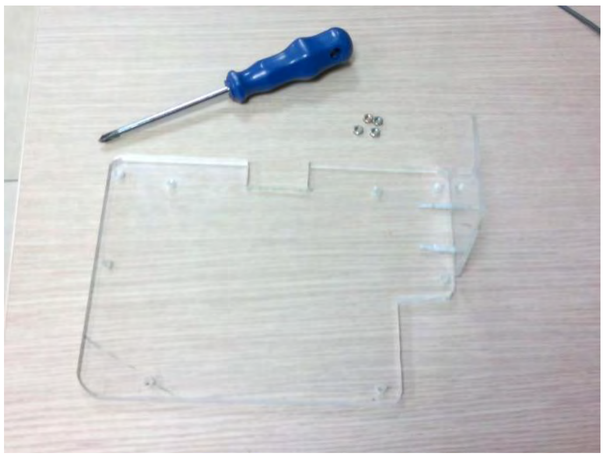
- *Male DC jack 2.5mm x 5.5mm x 10mm*

The center terminal on the power connector is wired to positive battery voltage, both pins 10 and 11 of the 25-pin male D-sub connector are used. The outer terminal of the power connector is connected to battery negative or ground, pin 25 and 14 of the 25-pin male D-sub connector is used. Approximately 35cm of power cable is needed.

The Alix Board requires an input voltage between 7 and 20V and draws consumption of approximately 12W.

4.4 Mounting Plate

Mounting the board on top of the Create requires some work. We used a costum designed tray, made from plexiglas, to hold the Alix Board. The tray had to be cutted, in order to reach the cords in the cargo bay. Also it had to be cutted at its side, in order to fit the power connector. We also made 4 holes to mount the board on the tray, and another 4 holes to mount the tray on the Create. Finally we took a small piece of plexiglas to mount the upward facing camera on it and glue the two pieces of plexiglas together.



4.5 Camera

We use a Logitech QuickCam for our robot.

5. Programming the robot

We use python scripts to determine the movements of the robot. Specifically, we used an implementation of the iRobot Create Open Interface, called *openinterface.py*. For the image recognition we used ARToolKit, a software library for building Augmented Reality(AR) applications.

5.1 iRobot Create

At the beginning, we should connect the serial cable between iRobot and a PC's serial port.

Openinterface

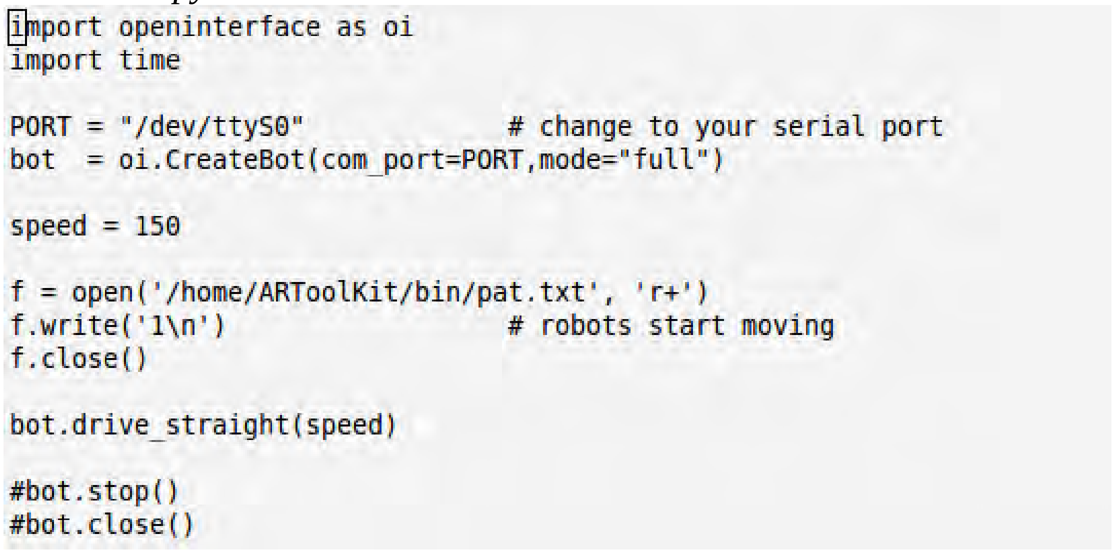
openinterface.py is a single file, functional implementation of the iRobot Create Open Interface specification. It's an easy way to get introduced into programming (Python is often used as a teaching language) and programming robots in particular (using the OpenInterface). This module, and its ancestor sci.py, have been used for a variety of purposes ranging from real robotics research to recreational hacking.

The installation of *openinterface.py* requires the following steps:

- Install Python2.4 or higher
- Install pyserial from http://pyserial.sourceforge.net/pyserial.html
- Copy the *openinterface.py* to your project directory.

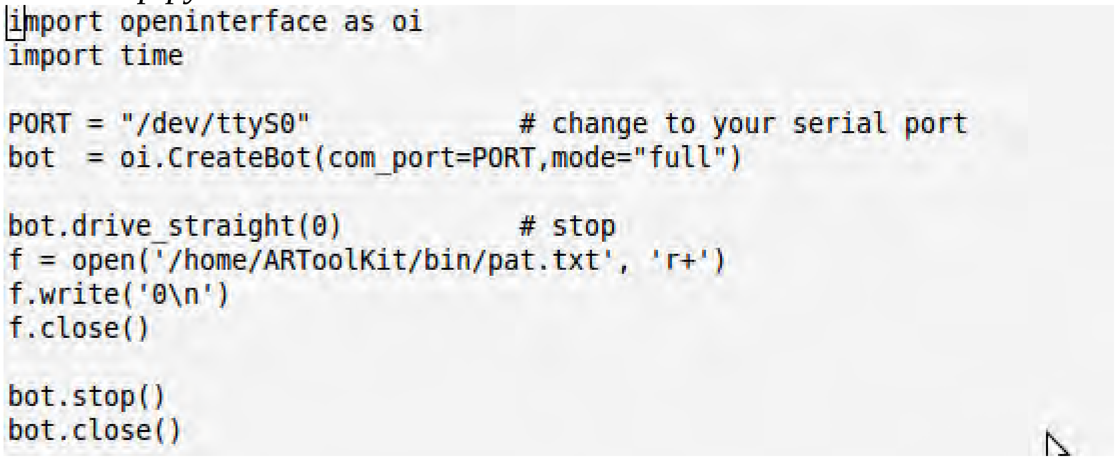
These are some python scripts we use to move iRobot:

- *start.py*



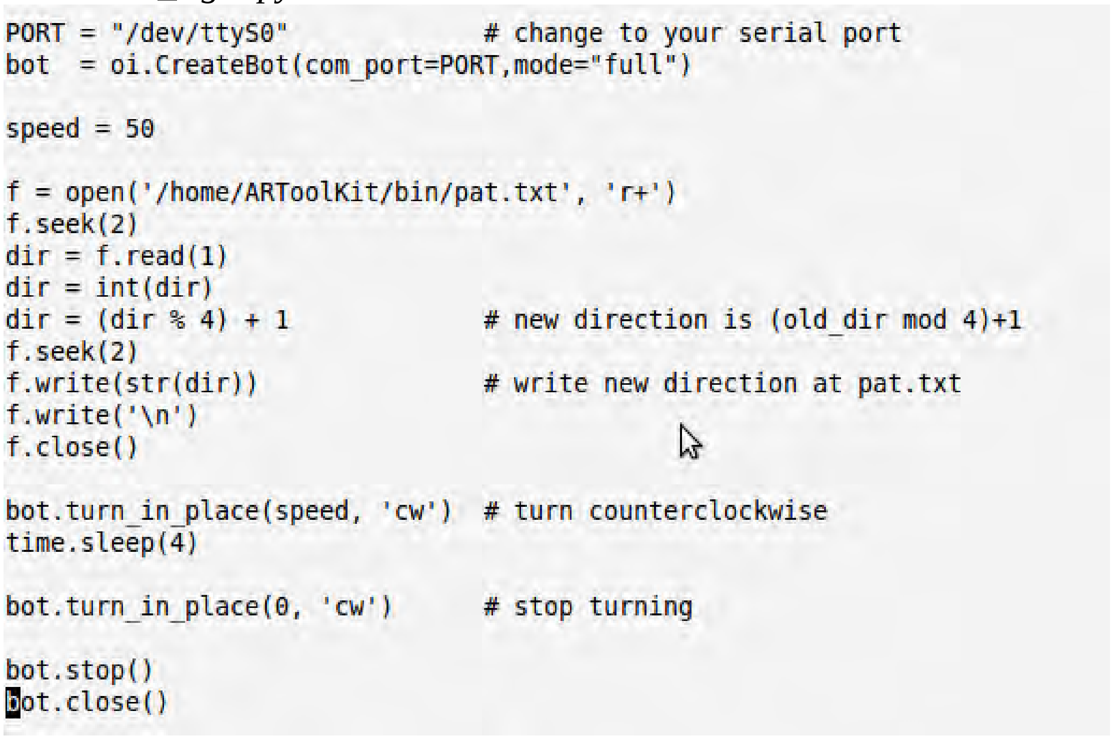
```
import openinterface as oi
import time

PORT = "/dev/ttyS0"              # change to your serial port
bot  = oi.CreateBot(com_port=PORT,mode="full")

speed = 150

f = open('/home/ARToolKit/bin/pat.txt', 'r+')
f.write('1\n')                   # robots start moving
f.close()

bot.drive_straight(speed)

#bot.stop()
#bot.close()
```

- *stop.py*



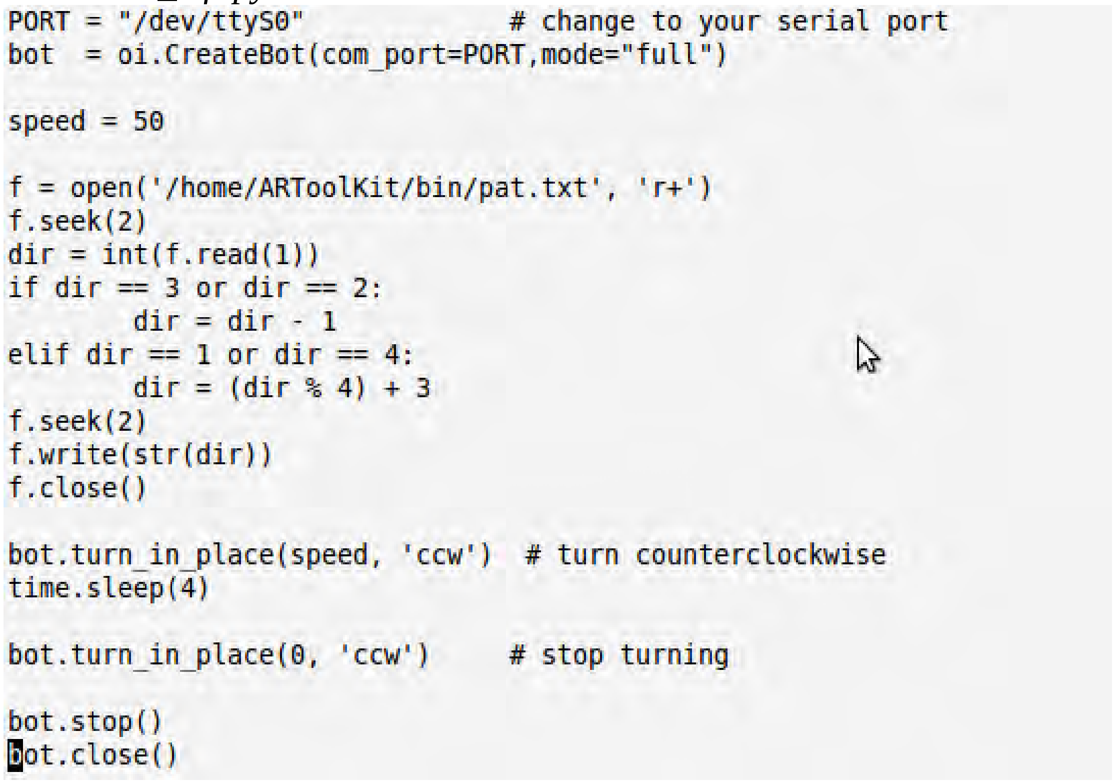
```
import openinterface as oi
import time

PORT = "/dev/ttyS0"              # change to your serial port
bot  = oi.CreateBot(com_port=PORT,mode="full")

bot.drive_straight(0)            # stop
f = open('/home/ARToolKit/bin/pat.txt', 'r+')
f.write('0\n')
f.close()

bot.stop()
bot.close()
```

- *turn_right.py*



```
PORT = "/dev/ttyS0"             # change to your serial port
bot  = oi.CreateBot(com_port=PORT,mode="full")

speed = 50

f = open('/home/ARToolKit/bin/pat.txt', 'r+')
f.seek(2)
dir = f.read(1)
dir = int(dir)
dir = (dir % 4) + 1             # new direction is (old_dir mod 4)+1
f.seek(2)
f.write(str(dir))               # write new direction at pat.txt
f.write('\n')
f.close()

bot.turn_in_place(speed, 'cw')  # turn counterclockwise
time.sleep(4)

bot.turn_in_place(0, 'cw')      # stop turning

bot.stop()
bot.close()
```

- *turn_left.py*



```
PORT = "/dev/ttyS0"              # change to your serial port
bot  = oi.CreateBot(com_port=PORT,mode="full")

speed = 50

f = open('/home/ARToolKit/bin/pat.txt', 'r+')
f.seek(2)
dir = int(f.read(1))
if dir == 3 or dir == 2:
        dir = dir - 1
elif dir == 1 or dir == 4:
        dir = (dir % 4) + 3
f.seek(2)
f.write(str(dir))
f.close()

bot.turn_in_place(speed, 'ccw')  # turn counterclockwise
time.sleep(4)

bot.turn_in_place(0, 'ccw')      # stop turning

bot.stop()
bot.close()
```

5.2 Alix Board

5.2.1 CF card

First we need to copy our file system to the CF card. We connect the CF card to a card reader and then, do the following:

```
$sudo fdisk –l                // to see a list of our disks

$sudo fdisk /dev/sdb          // to enter our Cf card’s menu
```

-m for help

-o for a new empty DOS partition table

-n to add a new partition
--primary partition
--partition number 1
--we hit enter to the rest, for default values

-a to set the boot flag
--partition number 1

-w to save the new partition table

- formatting our new partition:

```
$sudo mkfs.ext2 /dev/sdb1

$sudo tune2fs -c 0 /dev/sdb1       // never do fsck on that filesystem
```

- copy to CF card

```
$sudo mkdir /mnt/cf                // the mounting point

cd voyage_folder/usr/local/sbin
$sudo ./voyage.update
```

-select 1
-select the voyage folder

-select 3 for target disk
--select disk: /dev/sdb
--select partition: 1
--select mounting point we made earlier

we are ready to copy voyage at CF card
-select 6
-select y

5.2.2 Booting

- Install our bootable CF card to the Alix Board.
- Connect serial port to a PC through null modem cable(RXD/TXD crossed over). Set terminal emulator to 38400 8N1, flow control = none or xon-xoff.*
- Connect Ethernet as needed.
- Connect a 18V DC power supply to the DC jack. Power supply sould be able to supply at least 12W for some margin. To avoid arcing, please plug in the DC jack first, then plug the adapter into mains.

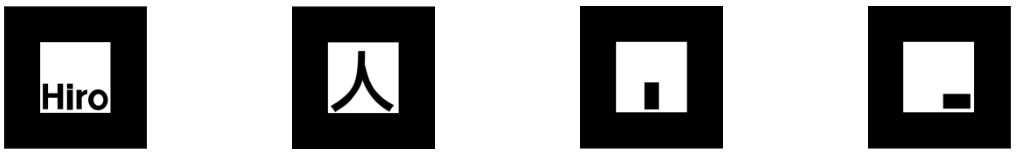
* To communicate through serial port, we used minicom

The board should now power on. All three LEDs will light during BIOS POST, then the system will try to boot, and turn off two of the LEDs. You should see tinyBIOS startup messages, memory size, CF disk geometry on the serial console.

5.2.3 ARToolKit

ARToolKit is a C language software library that lets programmers easily develop Augmented Reality applications. Augmented Reality (AR) is the overlay of virtual computer graphics images on the real world, and has many potential applications in industrial and academic research.

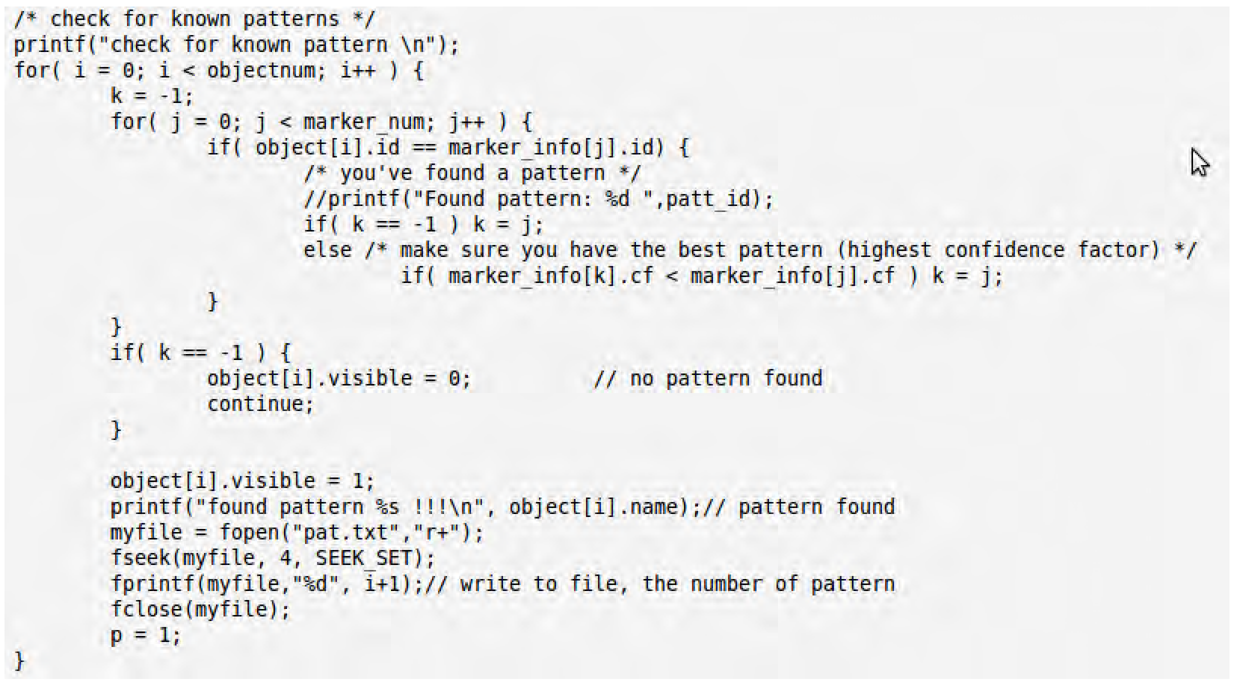
Using ARToolKit we recognize 4 patterns positioned on the ceiling and give the appropriate command to iRobot. The 4 patterns are:



Each time we found a pattern, its number is printed in a file called *pat.txt*.

This is the code we use to recognize the patterns:

loadMultiple.c

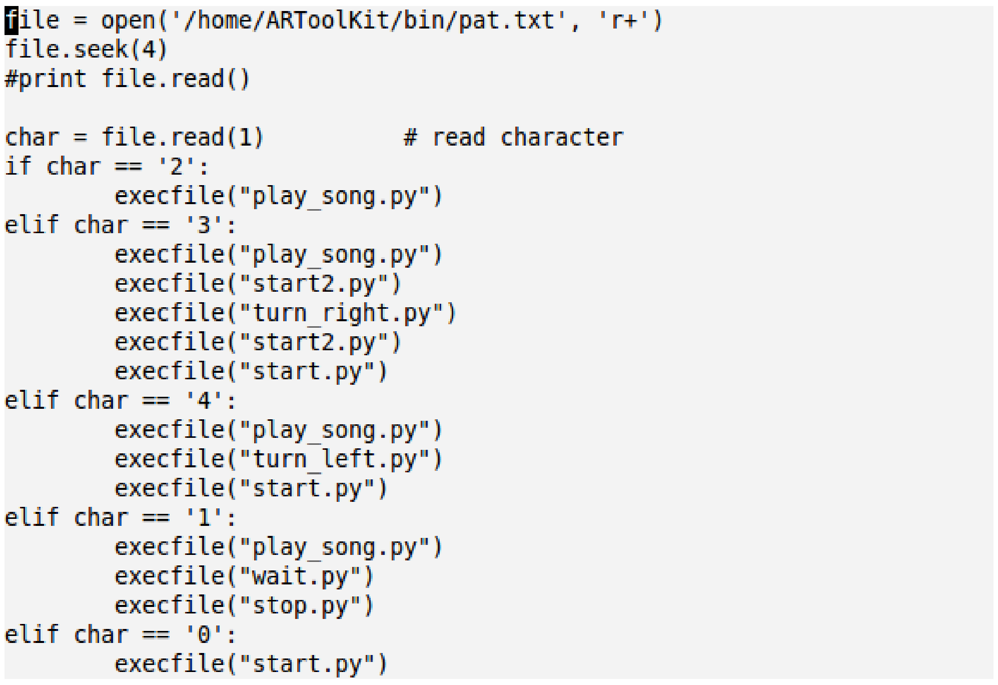


```
/* check for known patterns */
printf("check for known pattern \n");
for( i = 0; i < objectnum; i++ ) {
        k = -1;
        for( j = 0; j < marker_num; j++ ) {
                if( object[i].id == marker_info[j].id) {
                        /* you've found a pattern */
                        //printf("Found pattern: %d ",patt_id);
                        if( k == -1 ) k = j;
                        else /* make sure you have the best pattern (highest confidence factor) */
                                if( marker_info[k].cf < marker_info[j].cf ) k = j;
                }
        }
        if( k == -1 ) {
                object[i].visible = 0;          // no pattern found
                continue;
        }

        object[i].visible = 1;
        printf("found pattern %s !!!\n", object[i].name);// pattern found
        myfile = fopen("pat.txt","r+");
        fseek(myfile, 4, SEEK_SET);
        fprintf(myfile,"%d", i+1);// write to file, the number of pattern
        fclose(myfile);
        p = 1;
}
```

5.2.4 Combining python scripts with ARToolKit

We made another python script, *reader.py*, to read the file *pat.txt* and according to the pattern that is recognized, navigates the robot.

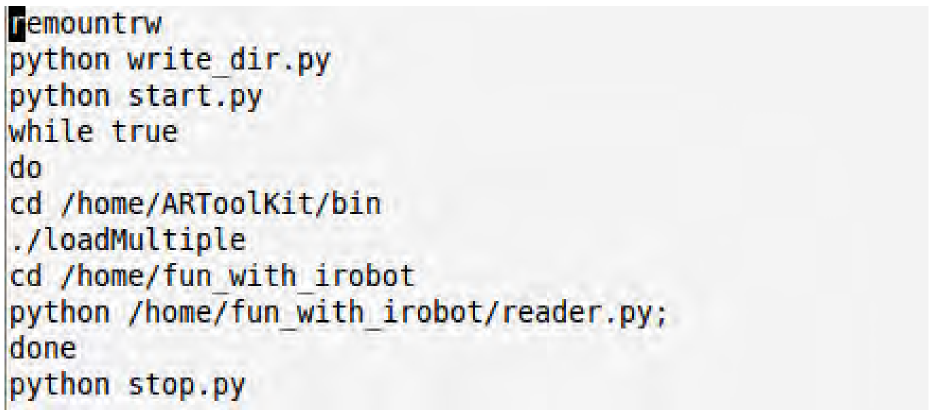
reader.py



```
file = open('/home/ARToolKit/bin/pat.txt', 'r+')
file.seek(4)
#print file.read()

char = file.read(1)         # read character
if char == '2':
        execfile("play_song.py")
elif char == '3':
        execfile("play_song.py")
        execfile("start2.py")
        execfile("turn_right.py")
        execfile("start2.py")
        execfile("start.py")
elif char == '4':
        execfile("play_song.py")
        execfile("turn_left.py")
        execfile("start.py")
elif char == '1':
        execfile("play_song.py")
        execfile("wait.py")
        execfile("stop.py")
elif char == '0':
        execfile("start.py")
```

Finally, *script.sh* take over the process to combine and synchronize the two discrete parts, movement and image recognition.

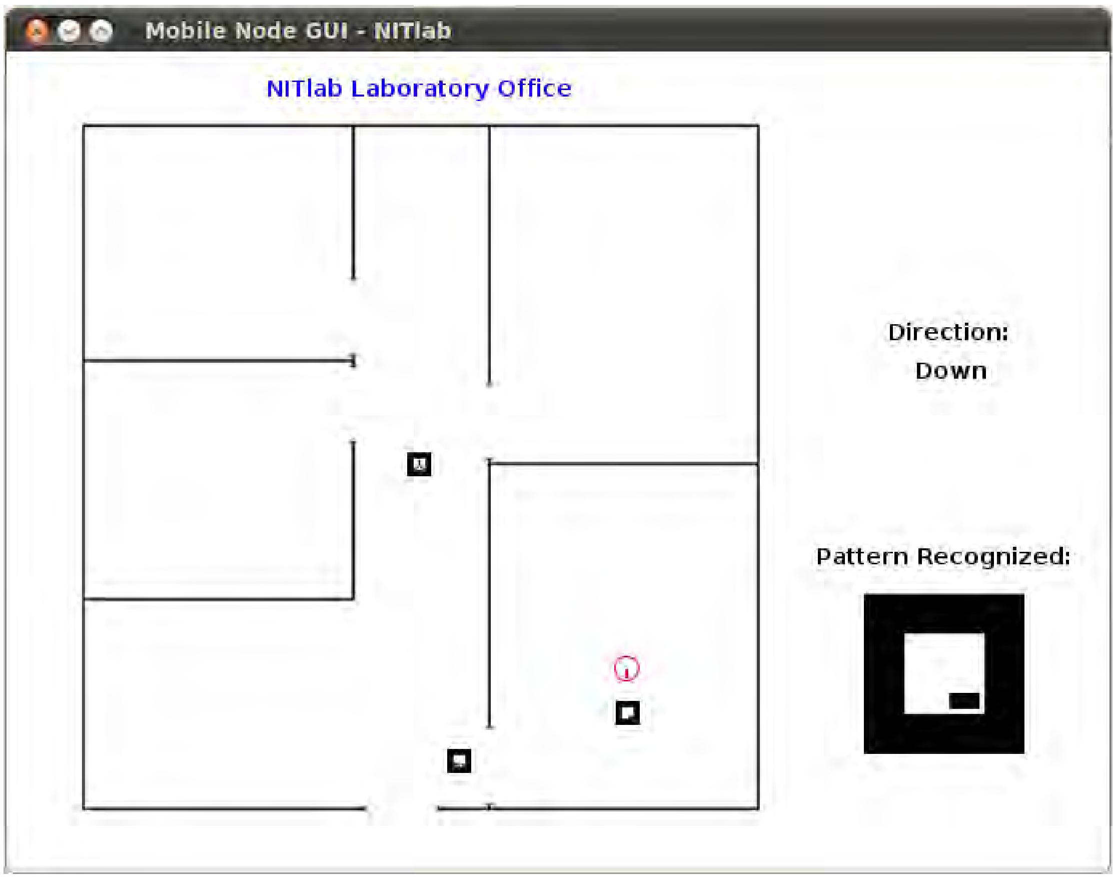
script.sh



```
remountrw
python write_dir.py
python start.py
while true
do
cd /home/ARToolKit/bin
./loadMultiple
cd /home/fun_with_irobot
python /home/fun_with_irobot/reader.py;
done
python stop.py
```

6. Mobile node GUI

This is the gui for running experiments.



We can see a layout of NITlab laboratory office, where the robot moves around according to the pattern recognized. At the right side of the gui, the direction and the pattern recognized is displayed. Each time robot changes direction, the direction displayed becomes red for a while, and each time robot recognizes a pattern, the pattern blinks, to inform that something has changed.

7. Future work

There are many more things to do on our mobile node in the future.

Firstly, we could train the camera to recognize more patterns, which means more complexity for robot’s movement, thus for our experiments.

Secondly, we could implement orientation tracking with ARToolKit library. The ARToolKit video tracking libraries calculate the real camera position and orientation relative to physical markers in real time. Therefore, each time a pattern is recognized we will be able to know the exact angle that camera “sees” the pattern.

Finally, we can take advantage of our mobile node at NitLab testbed, to study mobility in wireless experiments.

REFERENCES

[1] *Mobile Emulab: A Robotic Wireless and Sensor Network Testbed*. David Johnson, Tim Stack, Russ Fish, Daniel Montrallo Flickingery, Leigh Stoller, Robert Ricci, Jay Lepreau.

[2] *Mobility in Wireless Networks*. Christian Schindelhauer.

[3] *Mobility Increases the Capacity of Ad Hoc Wireless Networks*. Grossglauser, M., and Tse.

[4] *iRobot Create Open Interface Manual*
http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf

[5] *iRobot Create Owner's Guide*
http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf

[6] OMF Enabled-Robot at NICTA Research Centre.
http://omf.mytestbed.net/projects/robot/wiki/

[7] *The SmURV Robotics Platform*. Jonas Schwertfeger, Daniel Hartmann, Mark Buller, Odest Chadwicke Jenkins
http://robotics.cs.brown.edu/projects/smurv/

[8] *PC Engines – alix 2d2*: http://www.pcengines.ch/alix2d2.htm

[9] *Pyserial*. http://pyserial.sourceforge.net/pyserial.html

[10] *ARToolKit*. Hirokazu Kato, Mark Billinghurst, Ivan Poupyrev.