GotABot

User Guide version 4.16.0

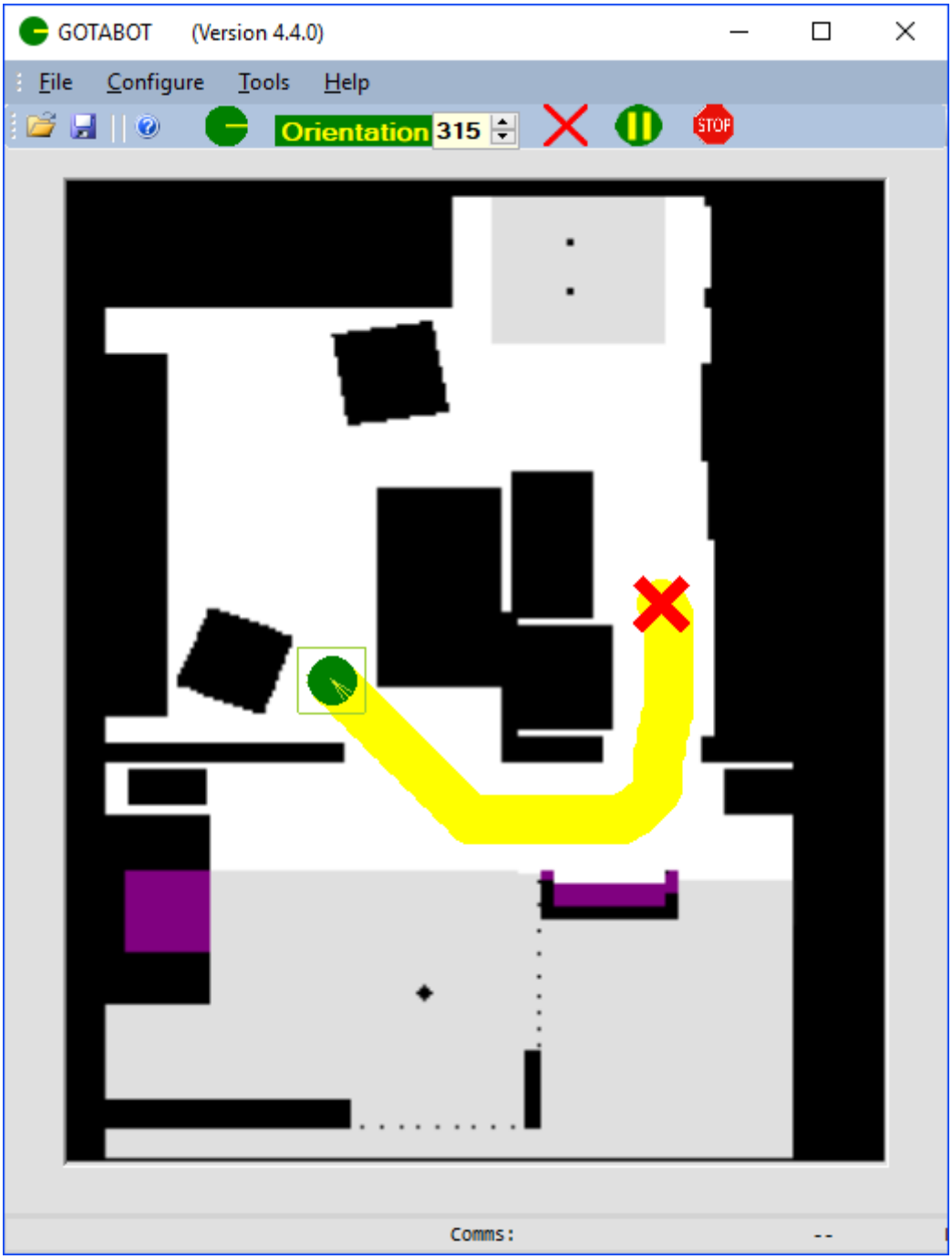


Table of Contents

1. Introduction

So you've got a robot and need some software to make it do stuff. Perhaps GotABot can help with that. Using a map, it can plan a route and direct your robot to a desired destination. With odometry and a camera , Kinect or laser scanner, GotABot will track the robot and correct its position and trajectory so as to stay on course.

GotABot also supports optional enhancements such as an arm, bumper switches, Kinect or URG04LX Laser Range Finder.

Additional information and the latest version of the software may be found on the website: http://gotabot.weebly.com/.

Important Safety Information

Although great effort has been taken to debug GotABot, it is a "Use at your own risk" type program. Please use common sense and follow all safety instructions provided by the robot manufacturer.

2. Getting Started

Place the robot on the map by:
1) Click this icon
2) Then click the map at the desired location

Make the robot go to a destination by:
1) Click this icon
2) Then click the map at the desired location

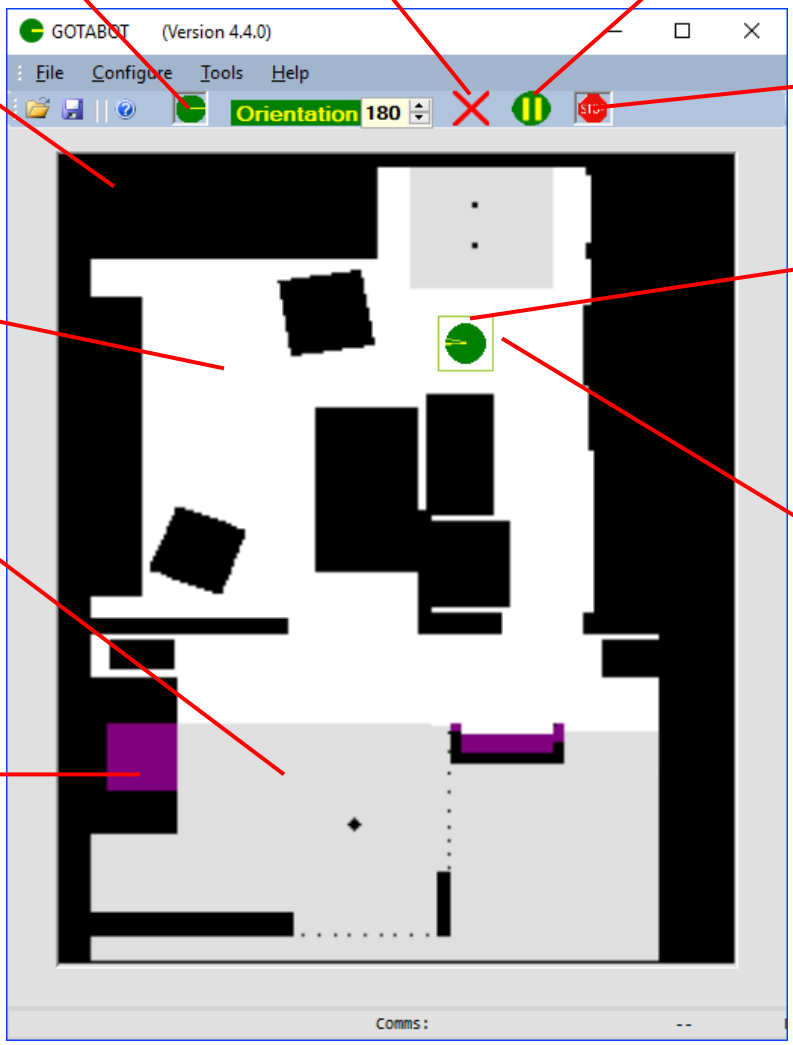
Pause the robot's movement. Re-click to resume movement.

Black (Red=0, Green=0, Blue=0) indicates solid objects and areas where the robot can't go.

White(Red=255, Green=255, Blue=255) indicates areas where the robot is free to move.

Grey (Red=223, Green=223, Blue=223) indicates areas where the robot can try to go if necessary, but probably shouldn't.

Purple (Red=128, Green=1, Blue=128) indicates areas which Range Finders or sensors indicate as empty but which are in fact solid objects or areas where the robot should not go.



Stop the robot's movement.

This is the robot. The bright yellow line indicates the direction, or orientation, it is facing.

Yellow-Green lines indicate the uncertainty region of the robot's pose.

Figure 1 Overview

QuickStart

Unzip the Download

Choose Your Robot

If Using an Evolution Robotics ER1 robot

1) Configure GotaBot and the ER1
2) Choose the localization method and configure

If Using a Heathkit Hero-1 robot

1) Use the Heathkit Hero-1 setup guide

If Using an Anki Vector robot

1) Use the Anki-Vector setup guide

Create a map

1) Create a map

Choose Your Sensor

If Using a Kinect or URG04LX Laser Range Scanner or a Heathkit Hero 1 robot

1) Configure the Scanner.
2) Configure various variables
3) Operate

If Using a Camera and LandMarks

1) Setup some Landmarks
2) Configure various variables
3) Operate

Then What?

Once basic operation is achieved the user may explore this manual for additional capabilities and options.

TIP: Easy Localization

Localization using a Laser Scanner or Kinect is easier, faster and more accurate than localization using a camera and landmarks.

System Configuration and Installation of GotABot

System Configuration

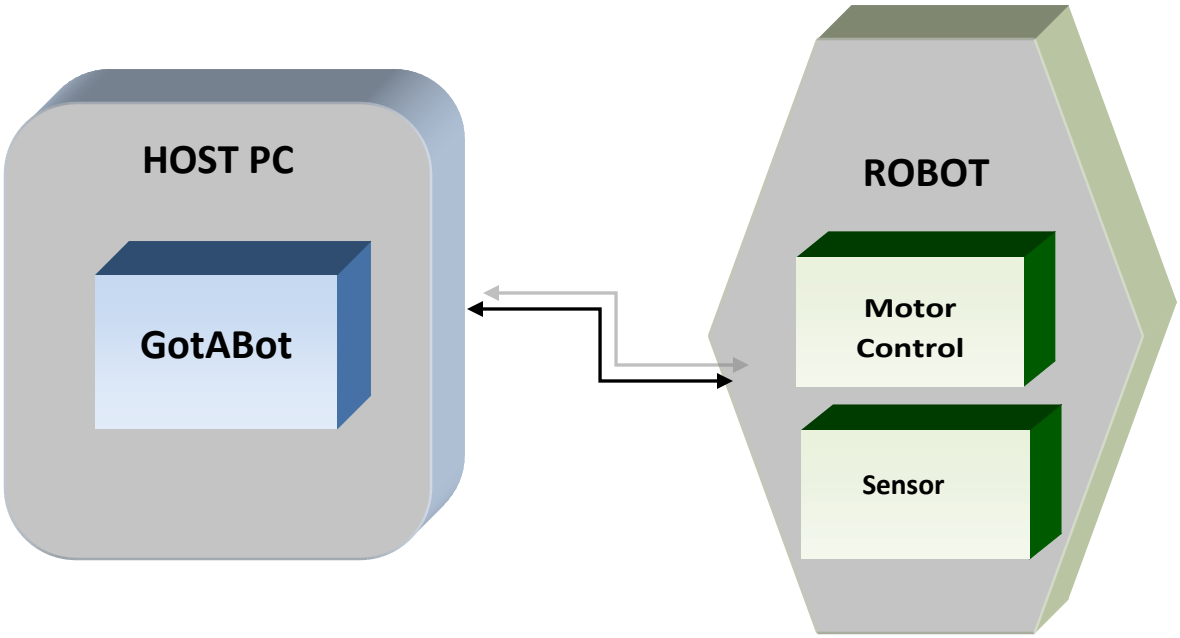
Before installing GotABot, one should first decide how to configure one's robotic system. There are a number of factors to consider such as:

1) The robot which is to be controlled
2) Will GotABot act as a standalone controller or will it be controlled by a higher level program.
3) Will the robot use a Kinect or laser scanner or a webcam to determine its position.

These and other considerations will be discussed in subsequent chapters.

Simple System Configuration

The simplest system configuration is as shown below. The Host PC may be separate from the robot and communicate to it via a wireless link or it might be on or part of the robot itself. This is the easiest configuration and should,be used first if for no other reason than to familiarize oneself with GotABot operation.



Other System Configurations

Although the previous 'Simple System Configuration' works well, there are other configurations which may be more optimal for some applications:

1) If you wish to add additional behaviors, customize GotABot's operation, or allow a separate program to use all or portions of GotABot's functionality then see the chapter "API".
2) A Kinect or URG04 Scanner is easier to setup and can provide better performance than image recognition of Landmarks. For even more reliable results, both Scanner and Landmark recognition may be used simultaneously.

Installation

For most applications, GotABot is installed on a host computer mounted on your robot.

The absolute minimum requirements of the computer are:

- Pentium® III class, Intel® Celeron®, or AMD processor, or better - 1000 MHz or faster
- Windows XP or better (Windows XP must be updated with Net Framework)
- 256 MB RAM
- 150 MB of free Hard Disk Space

Naturally, for zippier performance, a more modern computer is desirable:

- Intel i3, or better
- Windows 10
- 4 GB RAM
- 150 MB of free Hard Disk Space

GotABot is a portable application and it may be unzipped or copied to any convenient directory.

3. Creating a Map

Maps are BMP files representing the environment the robot will operate in. They can be created by most graphics programs including Microsoft Paint.

- **Black** (Red=0, Green=0, Blue=0)areas represent walls, furniture and "out of bound" areas.
- **Grey** (Red=223, Green=223, Blue=223) represents areas which may be occupied or areas where the robot should not travel through unless it can find no other route.
- **White** (Red=255, Green=255, Blue=255) represents open space through which the robot is free to move.
- **Purple** (Red=128, Green=1, Blue=128) indicates areas which Scanners indicate as empty but which are solid objects or areas where the Robot should not go.

Scale

GotABot uses a map scale of 1 grid square = 1 square inch.

Creating Maps With Microsoft Paint

1) *Measure the actual area you want mapped – be accurate.*
2) *GotABot uses a map scale of 1 grid square = 1 square inch. As an example, a 10 x 12 foot space requires a map of 120 x 144 grid squares.*
3) *Start Paint (Start -> Run..->'mspaint')*
4) *Set the Map Size (Press Ctrl+E then set the map Width and Height)*
5) *Set the Map Units (Press Ctrl+E then set the map units to 'Pixels')*
6) *Enlarge the map (Press Ctrl+PgUp)*
7) *Show a grid (Press CTRL+G)*
8) *Using the grid and the mouse position indicator in the bottom-left, draw the map.*

As an example, this is a map that I use:

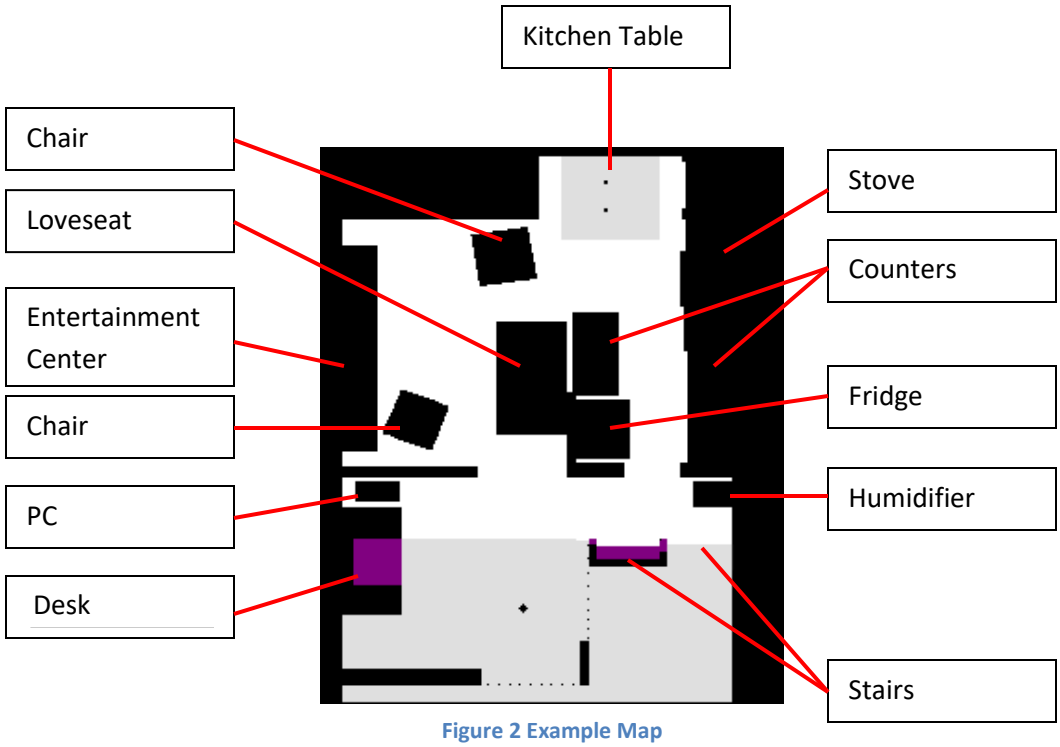


Figure 2 Example Map

…but shouldn't the robot make the map?

Well, it can (see the chapter on SLAM), but unfortunately, making a high accuracy map is a decidedly non trivial problem. Robot generated maps are especially problematic when the only measuring instruments are a cheap webcam, dodgy odometry and low cost range sensors. All in all, the best maps are drawn by old fashioned humans.

4. Scanner Configuration

GotABot provides interfaces to a Microsoft Kinect (first generation) or to a URG04 Laser Range Finder. Either will provide excellent localization and collision prevention. Localization using one of these Scanners is in many ways superior to using visual Land Marks as it is easier to configure, more accurate and localizes faster.

Configuration

Configure operation by opening the Scanner Configuration window (click *Configure->Scanner*).

The Offset fields represent the Scanner's position on the robot relative to the robot's center (typically centered on the drive wheels). Click the 'Continuous Scan' box and adjust the Calibration sliders to correct any errors in the range measurements.

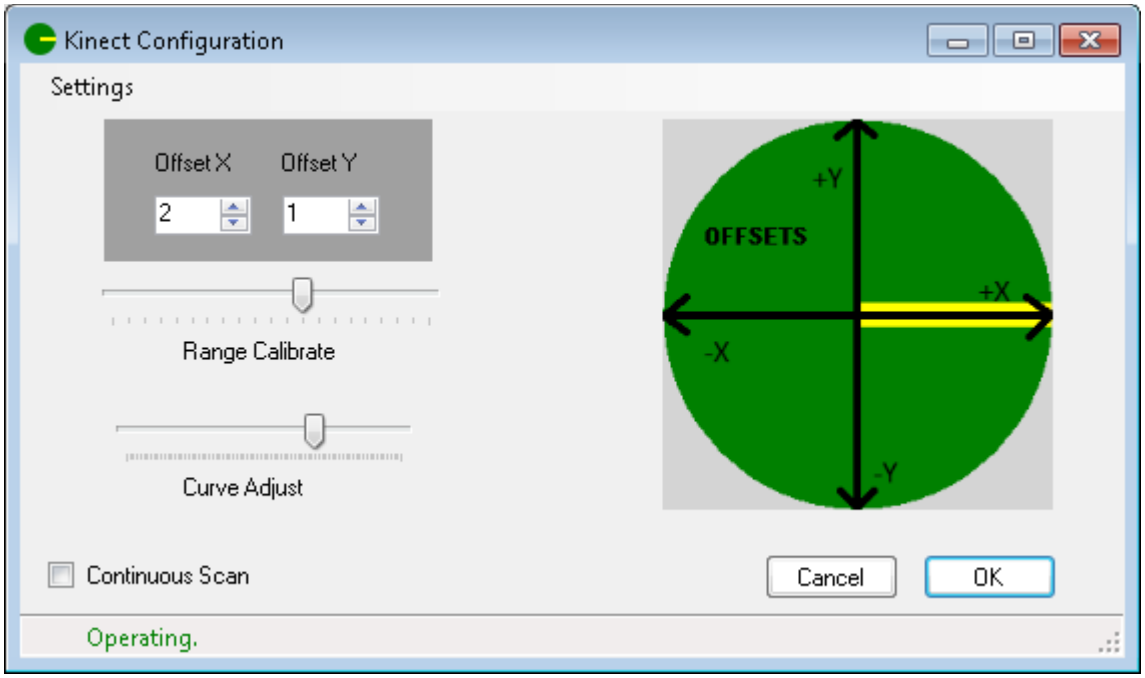


Figure 3 Scanner Configuration Window

TIP: Kinect Installation

The Kinect sensor requires an external power source beyond that applied through the USB port. Driver Version 1.8 is recommended. Install the SDK - not just the drivers.

5. Landmark Configuration

In addition to the Scanner interfaces, visual Landmark recognition may be used for determining the robot's position. Landmarks are user selected images that the robot can see through its webcam so as to recognize and measure range and direction to. Knowing the Land Mark's location and the estimated robot's position and orientation, GotABot uses simple geometry to determine where the robot is located.

For detection of Land Marks, GotABot provides interfaces to the Evolution Robotics RCC image recognition program (Available for download from the web).

RCC

This application is provided with the Evolution Robotics ER1 robot. Included in this application is a powerful image recognition utility suitable for identifying many types of images and useful for more than just ER1 control .

Install RCC in *c:\Program Files\ER1* or *c:\Program Files (x86)\ER1*

To configure this RCC application:

i) click *Settings->Remote Control*

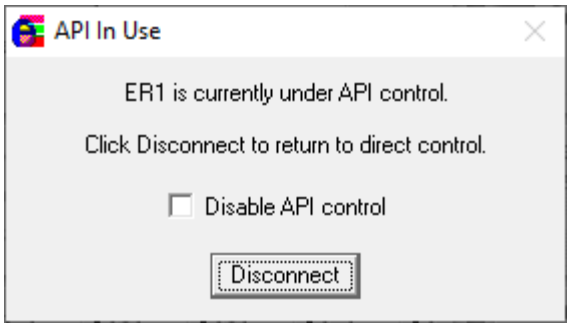
ii) check *'Allow API control of this instance*'

iii) set *Port* = 9000

iv) *Password* = blank

(In the main GotABot window click *Configure-> Landmarks.* In the Landmark Configuration window click *Misc->Enable RCC.*)

If setup correctly, the RCC will display the following Window:



Establishing a Land Mark is realized by:

1) Select the Landmark
2) Capture an image of it
3) Enter it in the GotABot database

Selection of Land Mark Images

Without a doubt, the selection of good Land Marks is the most crucial component to ensuring adequate localization. A few well chosen Land Marks will be the equal of dozens of randomly chosen images of a typical home environment.

A good Land Mark will be:

1) Unique
2) Easily recognizable by the robot's image processing algorithm
3) Useable over a wide range of distance and angles

In practice, 2D images such pictures, posters and playing cards are excellent choices. When first setting up, it would be well advised to temporarily attach a few of these to strategically located spots. Later, they can be replaced or augmented by a series of normal room images.

Figure 4 Example of a Highly Recognizable Land Mark

Tips For Good Land Marks

1) A good Land Mark should contain a number of high contrast features which the robot's algorithm can easily discern. Silhouette, or black and white, images are highly recommended!
2) Land Marks are most robust when they are in a single plane such as a picture, poster or playing card. Using such a Land Mark, the distance from robot to Land Mark can be estimated accurately over a wide range of distances and angles.
3) If Landmark images containing various features at various distances must be used, the valid region over which distance can be accurately measured will be relatively shallow. In such situations, it is best to make multiple Land Mark images, each at a slightly different distance.
4) Land Marks will yield more accurate range information when the orientation of the robot to Land Mark is close to that of the original image. If the Land Mark will be seen over a wide range of angles then create multiple Land Marks by capturing images at different angles.
5) Images which primarily consist of repeating features such as sets of drawers, slatted doors, patterned wallpaper or furniture, railings, etc, do not make good Land Marks.
6) Land Mark images which contain unwanted objects may be edited in Paint. Usually, the desired details may be easily isolated by overwriting everything else with grey rectangles.
7) Good locations for Land Marks include ends of corridors, doorways and any place where careful maneuvering is required to pass through tight spaces.

8) Ideally, the Land Mark should be at the same height as the camera.
9) Landmarks may not be recognized if they are placed against a 'busy' background. In such cases, a blank border around the land mark will improve recognition (i.e. if your landmark is, say, a picture attached to a patterned background then insert a slightly larger blank sheet of paper between the picture and the background)

The Trouble With Land Marks

As a rule of thumb, there should be at least one Land Mark visible at any location the robot is likely to stop. With one Land Mark, the robot's pose (i.e. – its position and orientation) can usually be calculated accurately, but sometimes, especially when the robot's pose is not accurately known, errors will occur and the pose will be *shifted*. Usually, these errors will be corrected with time and further localizations, but sometimes the snowball accelerate downhill, the reported pose drifts ever further from reality and the robot runs into a wall.

The treatment for this error condition is simple – add more Land Marks. Make sure that wherever the robot is likely to look, there is a good Land Mark. If there are two Land Marks visible everywhere then so much the better. Errors introduced by one Land Mark will be corrected by the others.

Accuracy Counts!

When establishing a Land Mark, make sure that the center of recognition is accurately transcribed in the Land Mark Edit form. It is probably necessary and certainly advisable to check where the image recognition software thinks the center is.

Having established where the actual center is, make sure that the range to this point is also correct.

The Edit Land Mark Window

Land Marks are entered through the Edit Land Mark window (In the main window click *Configure->Landmarks.* In the Landmarks Configuration window click *LandMarks*.)

. Remember to save your settings after making change s (*Click File->Save Settings*).

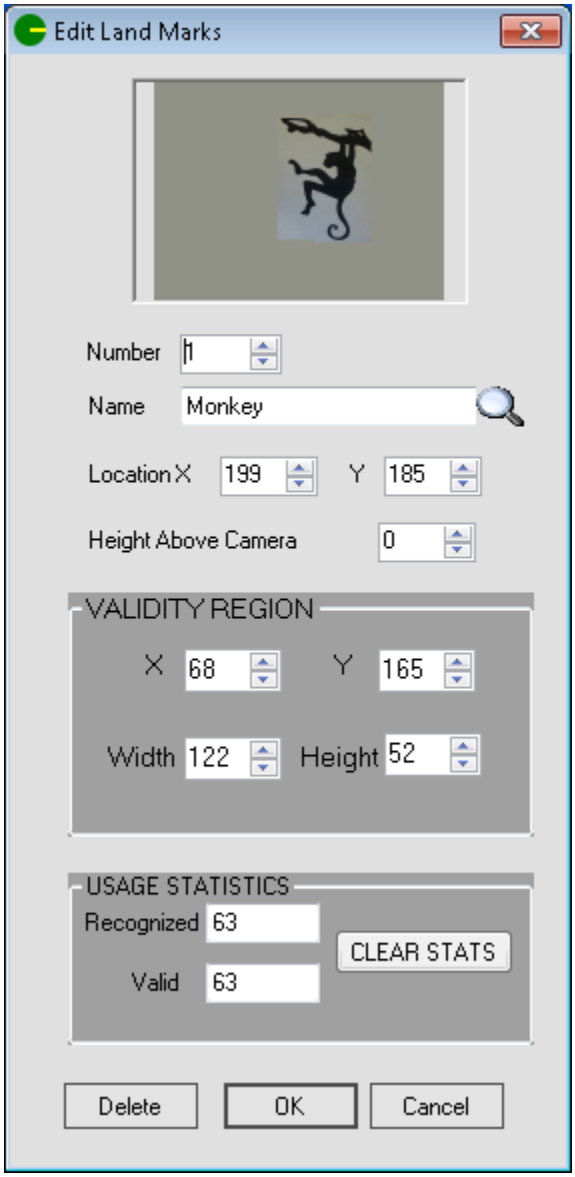


Figure 5 Edit Land Mark Form

Land Mark Number

Each Land Mark has a unique number. As the number field is changed, the corresponding Land Mark and its Validity Region are plotted on the map.

Land Mark Name

The Land Mark Name field must contain the exact name reported by the image recognition program.

Search Land Mark Name

Pressing the Magnifying Glass icon beside the Name field allows a search of the Land Mark database by name.

Land Mark Location

This is the location of the center of the Land Mark. It may be entered by:

1) Click on either of the Location X or Y up / down entry box.
2) Click on the map at the location of the Land Mark.

Height Above Camera

This is an optional field, but it can help to calculate a more accurate range to a LandMark when the LandMark is not at camera level. This is typically useful when the camera is near ground level and the LandMark is, say, a picture hung at a much higher eye level.

Land Mark Validity Region

To lessen the chance of harm due to false Land Mark recognitions, a validity region is required for each Land Mark. This region restricts the area and robot orientation within which the recognition will be judged to be valid.

The validity area is a rectangle and is entered by:

1) Click on either of the Validity Region X or Y up / down entry box.
2) On the map, position the mouse at one of the Validity Region's corners, press the left button and draw a rectangle - Top Left to Bottom Right. Release the button.

Land Mark Usage Stats

These values are generated by the GotABot program and provide information on the usefulness of each Land Mark. Low Recognition or Validity numbers may be an indication that the Land Mark information is incorrect or that a better Land Mark is required.

Break It Down For Me. How Do I Actually Setup A Landmark?

(Many steps but it's not difficult)

i) In the RCC application:

Ensure that RCC is not connected to GotABot (disable the RCC's API or shutdown GotABot)

In the Main Window - *Sight* section click *Add*

This will open an *Add Object* window - Click *Browse*

Select the image file that you wish to turn into a Landmark

In the *Add Object* window, fill in the *Name* field

Fill in a dummy value for *Distance From Camera*

click *OK*

Print out a copy of your image.

Aim your webcam at this image and confirm that it is recognized

Note the distance reported by RCC.

Click *Modify* under the *Sight* section

Change the *Distance From Camera* field so that the RCC reported distance matches the actual camera to landmark distance

ii) In GotABot's Landmark Configuration Window:

click *Landmarks*

This will open the Edit Landmarks Window

Enter the name of the Landmark - (must be the same as the RCC name)

If RCC is installed in *c:\Program Files\ER1* or *c:\Program Files (x86)\ER1* then you should see an image of your landmark

Fill in the *Location* and *Validity Region* fields. The *Validity Region* is the map area where RCC recognitions will be used (helps reject false recognitions)

In GotABot's main window, click *File->Save*

6. Additional GotABot Configuration

A number of other parameters may also be set.

Robot Selection

Selection of a robot to control is achieved through this option (*Click Configure->Bot->Robot Select ->* and select your robot.

Pose

The robot's pose may be set with this option (*Click Configure-> Pose*). A robot's pose is a series of parameters which describes the robot's position on a map, the direction it is facing, and the uncertainties in each of these parameters.

Settings

GotABot's configuration setting may be saved or reloaded by selecting the "File" menu Settings options.

Robot Diameter

The robot's diameter may be set with this option (*Click Configure->Bot->Robot Diameter*). The Robot Diameter not only determines the smallest openings the Robot can move through, but it also sets the distance the Robot will try to stay away from walls and other obstructions. Setting the diameter too large will restrict the Robot from certain openings and/or may cause the Robot to perform many small "stop, turn and go" corrections while navigating narrow spaces. Setting the diameter too small will result in the Robot colliding with the walls. It's a tradeoff.

Camera Resolution

This setting will define the camera's resolution. (In the main window click *Configure->Landmarks.* In the Landmarks Configuration window click *Camera->Camera Resolution.*) Click on the resolution desired.

Camera FOV

This setting will define the camera's diagonal FOV or Field Of View (In the main window click *Configure-> Landmarks.* In the Landmark Configuration window click *Camera-> FOV.*)

Camera Offset

This setting will define how far the camera is from the *turning* center of the robot (a positive value indicates that the camera is closer to the front of the robot than the turning center) (In the main window click *Configure-> Landmarks.* In the Landmark Configuration window adjust the *Offset X* field.) Note – the turning center is usually centered between the two drive wheels.

Velocity

The robot's linear velocity may be set with this option (*Click Configure->Bot->Velocity*)

Robot Acceleration

The robot's linear acceleration may be set with this option (*Click Configure->Bot->Acceleration*)

Robot Angular Velocity

The robot's angular velocity may be set with this option (*Click Configure->Bot-> Angular Velocity*)

TIP: Camera Selection / Configuration

Get the very best camera you can afford.

a) Choose a camera with a wide Field Of View. A minimum 60 degree FOV is recommended. More is better.

b) Choose a camera with good low light capability.

c) Choose the highest resolution the camera and PC speed will support. 1280 x 1024 works well. Even higher resolutions are better.

Robot Angular Acceleration

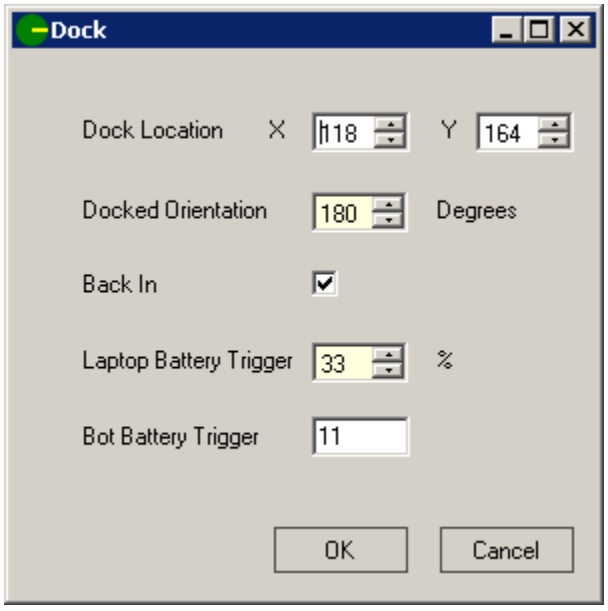
The robot's angular acceleration may be set with this option (*Click Configure->Bot-> Angular Acceleration*)

Docking Configuration

Docking is the process which allows the Robot to (automatically) recharge its batteries.

To configure automatic docking click *Configure->Dock:*

1) Dock Location is the ***Robot*** location when docked.
2) Docked Orientation is the Robot's orientation when docked.
3) Back In should be checked if the Robot must back into the Dock.
4) The Laptop Battery and Robot Battery Trigger fields will trigger an automatic docking operation when either of the respective battery levels falls below the trigger level



6 Docking Configuration Form

Localization Time

This setting may be used to control the length of time that GotABot will search for landmarks. Longer times will increase the number of times landmarks will be recognized (up to a maximum of 25 valid recognitions) thereby improving the localization accuracy. This setting may need to be increased if the image processing software is running on a slow PC or if other programs such as video servers are consuming the PC's time. (In the main window click *Configure-> Landmarks.* In the Landmark Configuration window click *Misc-> Visual Localization Time.)*

Mute

This setting prevents GotABot from speaking. (*Click Configure->Options->Mute*)

API Password

Sets the Password used to verify API commands. (In GotABot's main window Click *Configure->API Password*).

7. Operation

Software

By default, GotABot assumes that it is controlling an Evolution Robotics ER1 robot or a Heathkit Hero 1.

Operation Controls

Destination

The robot may be commanded to go to a destination by:

1) Click on the Tool Bar's Destination Icon ( )
2) Click on the map at the desired position

A yellow path (follow the yellow brick road) will be drawn on the map leading from the robot to the destination and the robot will be commanded to follow it.

Robot Position

The robot's position on the map may be changed by:

1) Click on the Tool Bar's Robot Icon ( )
2) Click on the map at the desired position
3) Click on the Tool Bar's Robot Orientation up/down controls (Orientation 0)
4) Adjust to the desired orientation

OR – Click *Configure-> Pose* and adjust as required.

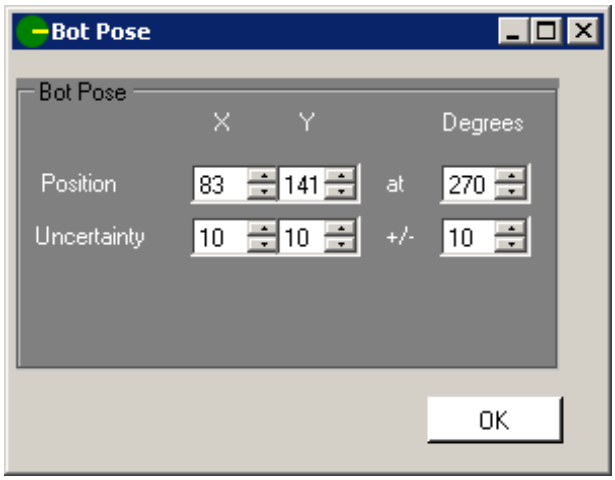


Figure 7 Robot Pose Form

Dock

Docking is the process which allows the Robot to (automatically) recharge its batteries.
Docking Configuration is described in Docking Configuration. To manually initiate a docking operation click *Tools->Dock* .

Localize

Selecting this option (*Click Tools->Localize*) will force GotABot to use Land Marks and / or a Scanner to determine its position and orientation

Simulated Movement

Selecting this option (*Click Configure->Options->Simulated Movement*) will prevent GotABot from communicating with the robot. Instead, robot movement is simulated on the map.

Stop

Double clicking on this Tool Bar icon (

) will force the robot to stop.

Debugging

When something screws up (and it surly will) or acts in an unexpected fashion, it is helpful to review the robot's actions. A log file of GotABot's operations is provided for this purposes *(click File->Open Log File*). This file, *GotABot_Log.txt*, may be consulted to review movement, localization and other events.

Manual Controls

As well as using the normal Menu selections, GotABot may be controlled by the following keys:

Arrow keys	Drive Forward, Left, Right and Back
W,A,S,D	Drive Forward, Left, Back and Right
U	Raise Head
J	Lower Head
T	Raise Arm
G	Lower Arm
F	Arm In
H	Arm Out
<Enter>	Speak Text

8. Voice Commands

Voice commands are provided via the web service IFTTT and through Google Home or Alexa. If using Google Home a stop command could be issued by saying:

"*Hey google trigger stop"*

All of the following commands must be preceded by the appropriate wake phrase, either "*Hey Google*" or "*Alexa*". If desired, the command syntax may be modified in IFTTT. In the following examples,. I added the words "*trigger*" or "*execute*" to many of the commands so that IFTT would be better able to recognize and correctly parse them. I chose "*execute*" as a way of instructing IFTTT the Command ends with a number. Except for "*go to*" commands, the other commands are preceded by saying "*trigger*".

The following is a partial list of commands which may be implemented.

Note that all of these Command phrases may be modified in IFTTT.

VOICE COMMAND	ACTION
" *trigger stop* "	Stop all motion
" *execute forward 12*"	Move forward twelve inches
" *execute back 8*"	Move backwards eight inches
" *execute turn right 45*"	Turn 45 degrees right
" *execute turn left 30*"	Turn 30 degrees left
" *trigger status* "	Report status
" *trigger localize*"	Perform a localization operation
" *trigger recharge*"	Go to dock and recharge
" trigger look up"	Look up *(requires Kinect or camera mounted on arm)*
" trigger look down"	Look down *(requires Kinect or camera mounted on arm)*
" *trigger look at me*"	Look directly at nearest face (*requires URG04*)
" *trigger my name is Bob*"	Tell the Face Recognizer your name
" *trigger remember my face*"	Train the Face Recognizer
" *trigger remember me*"	Train the Face Recognizer
" *trigger who am i*"	Say who is *seen* *(Azure recommended)*
" trigger open"	open the gripper
" trigger close"	close the gripper
" trigger home"	return the arm to its home position
" *execute out 10*"	Move gripper forward 10 mm
" *execute in 22*"	Move gripper back 22 mm
" *execute up 115*"	Move gripper up 115 mm
" *execute down 37*"	Move gripper down 37 mm
" *go to kitchen*"	Go to the kitchen *(define kitchen location in IFTTT)*
" *execute pick up 3*"	Drive to and pick up object 3
" *trigger batch test*"	Execute batch file 'test.bat'
" *trigger guard mode*"	Detect movement near Bot (*requires URG04)*
" *trigger photo*"	Take a photo
"trigger follow me"	Follows closest person or pet (requires URG04 or Azure)
"trigger describe person"	Describe person (*requires Azure)*
"trigger describe scene"	Describe scene *(requires Azure)*
"trigger identify objects"	Identifies objects *(requires Azure)*
" *trigger quit*"	Exit and close GotABot
" *trigger shutdown*"	Exit and shutdown PC

Voice Setup

Device Required

Make sure you have a Google Home or Alexa voice controlled speaker. Each provides a superior voice recognition service.

Router Configuration

Configure your router to forward Ports 9100 and 9400 to the PC running GotABot. On my router, P*ort Forwarding* is under *Network Settings -> Advanced. (*YMMV)

DDNS

If your internet provider does not assign static IP addresses then sign up with one of the free DDNS services. I use the Free Dynamic DNS from https://www.noip.com/remote-access.

GotABot API Configuration

Choose an API password (Click Configure->API Password)

IFTTT Account

1) Goto IFTTT
2) Sign in or sign up for an account.

IFTTT Configuration for Alexa

Alexa IFTTT App Example

Let's create a simple app that requests a status report:

1. Go to https://ifttt.com/create
2. Click *+This*
3. Choose Amazon Alexa
4. Choose "Say a specific phrase"
5. Type t*rigger status* in the "*What Phrase*" box
6. Click "*Create Trigger*"
7. Click *+That*
8. Click *webhooks*
9. Click *"Make a web request"*
10. *Set the URL box to "*http://*IPAddress*:9100/exec?pw=*Password*&script= status&end"
    replace *IPAddress with your IP address or ddns server; and Password with your API Password*
11. Set Method to "***POST***"
12. *S*et Content-Type to "**text/plain**"
13. Click "*Create Action*"
14. Click "*Finish*"

Alexa Status Test

When you say *"Alexa trigger status",* IFTTT will send an API command to your IP address at Port 9100. GotABot will then recite its status.

More Alexa Commands

Additional commands may be created by copying and modifying the above. Choose a command from the *API Commands* table in the manual. Simply change the trigger phrase to something appropriate and substitute the API command for status in *script= status*

IFTTT Configuration for Google Home

Google Home provides an easier yet more powerful interface than Alexa. Whereas each Alexa Applet can handle only one specific command, Google Home allows for more general purpose Applets, thereby cutting the total number of Applets required.

Google Home IFTTT Trigger Applet

Let's make a Trigger Applet which can handle most API commands.

1. Go to https://ifttt.com/create
2. Click *+This*
3. Choose Google Assistent
4. Choose "Say a phrase with a text ingrediant"
5. Type t*rigger $* in the "*What do you want to say*" box
6. Click "*Create Trigger*"
7. Click *+That*
8. Click *webhooks*
9. Click *"Make a web request"*
10. *Set the URL box to "*http://*IPAddress*:9100/exec?pw=*Password*&script= {{TextField}}&end"
    replace *IPAddress with your IP address or ddns server; and Password with your API Password*
11. Set Method to "***POST***"
12. *S*et Content-Type to "**text/plain**"
13. Click "*Save*"

Google Home IFTTT Execute Applet

Let's make an Execute Applet which can handle API commands which end with a number

1. Go to https://ifttt.com/create
2. Click *+This*
3. Choose Google Assistent
4. Choose "Say a phrase with both a number and text ingrediant"
5. Type execute *$ #* in the "*What do you want to say*" box
6. Click "*Create Trigger*"
7. Click *+That*
8. Click *webhooks*
9. Click *"Make a web request"*
10. *Set the URL box to " http:// IPAddress:9100/exec?pw= Password &script= {{TextField}}&xx= {{NumberField}}&end* "
    replace *IPAddress with your IP address or ddns server; and Password with your API Password*
11. Set Method to "***POST***"

12. Set Content-Type to "**text/plain**"
13. Click "*Save*"

Google Home IFTTT Go To Applet

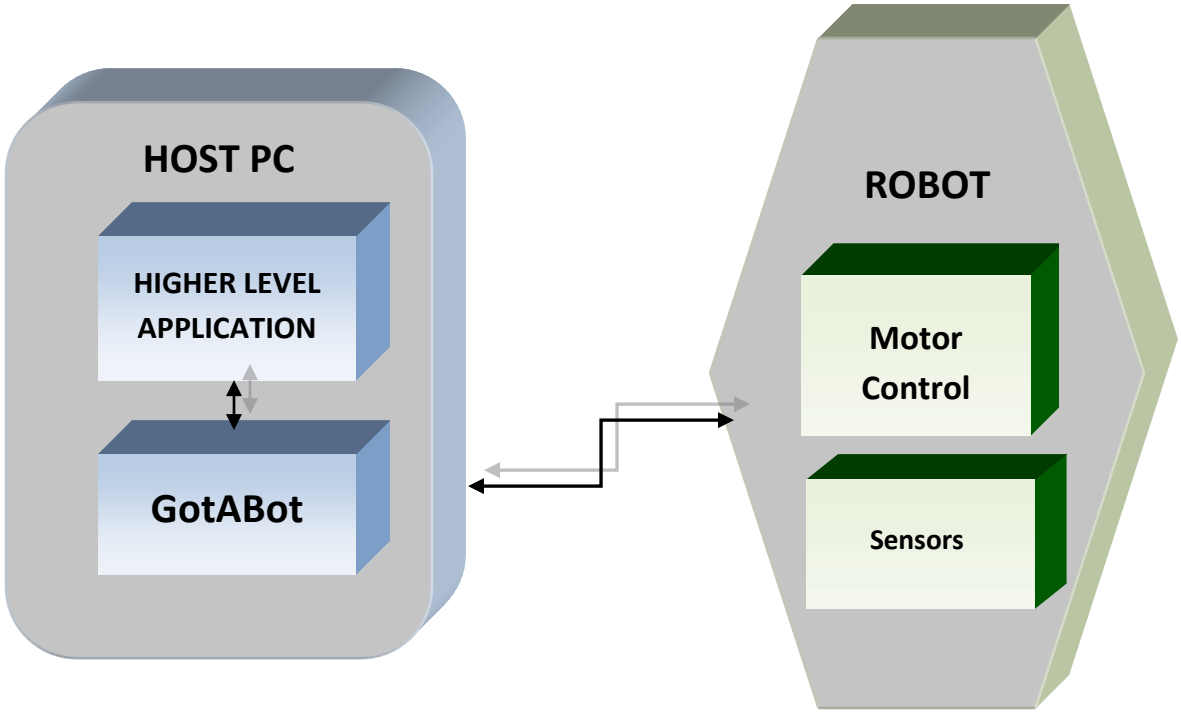
Let's make an example Go To Applet. This one instructs GotABot to go to the kitchen, which, in this example, is at coordinates 180,115. You will need to create a separate Applet for each destination.

1. Go to https://ifttt.com/create
2. Click *+This*
3. Choose Google Assistent
4. Choose "Say a simple phrase "
5. Type *go to kitchen* in the "*What do you want to say*" box
6. Click "*Create Trigger*"
7. Click *+That*
8. Click *webhooks*
9. Click *"Make a web request"*
10. *Set the URL box to " http://IPAddress:9100/exec?pw=Password&script=set&script2=destination&script2=destination&script3=180&script4=115&end* "
    replace *IPAddress* with your IP address or ddns server; and *Password* with your API Password
11. Set Method to "***POST***"
12. Set Content-Type to "**text/plain**"
13. Click "*Save*"

9. API

Up to this point, we have discussed how GotABot may be controlled through the GUI (Graphical User Interface) or via voice commands. This chapter will explore an alternate control method, the API (Application Program Interface). The API is accessed by opening a TCP/IP socket to GotABot and using simple text commands to control GotABot and / or any robots connected to GotABot. This method of providing control is common and widely used by many robot control programs.

An API is especially useful when we have a higher level program or script which provides additional functions or behaviors for the robot. Theoretically, one could create a stack of API enabled programs, each relying and building on lower level functions to achieve new heights of robot behavior.



GotABot contains two identical API interfaces (port 9100 or port 9400), either of which can be used by other programs or the user to pass commands or data to GotABot.

Accessing the API through Telnet

Accessing the API can be as simple as using a Telnet session. Although manual telnet control may have limited practical applications, it is very useful for debugging the commands and their sequences before implementing them in a script or program.

There are many different telnet programs but, as an example, here is how one could telnet to GotABot using the telnet program provided with Windows XP:

1) On the PC, click the **Start** button (lower left corner of your screen).
2) Click **Run** and type: *"telnet <ip address> <port>"*

Where the <ip address> is the ip address of the PC running GotABot
and <port> is 9100 for API #1 or 9400 for API #2
An example could be "*telnet 192.168.1.150 9100*"

3) Once connected, any of the API commands may be entered into the Telnet program.

TIP – Install Telnet Client for Windows 7, 8 or 10

As explained in http://technet.microsoft.com/en-us/library/cc771275(v=ws.10).aspx, the telnet client is not installed by default for Windows 7, 8 or 10.

TO INSTALL:

1. Open a command prompt window. Click **Start**, type **cmd** in the **Start Search** box, right click on *cmd.exe* and choose "*Run as administrator*"
2. Type the following command:

```
pkgmgr /iu:"TelnetClient"
```

3. Follow any prompts to install.

API Commands

<PW> stop
<PW> pause
<PW> turn right xx
<PW> turn left xx
<PW> forward xx
<PW> back xx
<PW> set destination xx xx

<PW> set pose xx xx xx xx xx xx
<PW> dock
<PW> localize
<PW> get status
<PW> get status full
<PW> get path xx xx

<PW> home *(requires arm)*
<PW> open *(requires arm)*
<PW> close *(requires arm)*
<PW> pick up xx *(requires arm or Vector)*
<PW> put down *(requires arm or Vector)*
<PW> pull xx *(requires arm)*
<PW> up xx *(requires arm or Vector)*
<PW> down xx *(requires arm or Vector)*
<PW> in xx *(requires arm)*
<PW> out xx *(requires arm)*
<PW> arm angles xx xx xx xx xx *(requires arm)*
<PW> set gripper xx xx *(requires arm)*
<PW> locate object xx *(requires arm)*
<PW> get snack xx xx xx xx *(requires Vector)*
<PW> get object xx xx xx xx xx *(requires arm)*

<PW> look up (requires arm or Kinect *or Vector*)
<PW> look down (requires arm or Kinect *or Vector*)
<PW> look at me (requires URG04)
<PW> remember my face
<PW> my name is **ssssss**

<PW> guard mode **ssssss** (requires URG04 and/or camera)
<PW> patrol xx xx **xx**
<PW> clean (requires Vector)
<PW> describe person (requires Azure)
<PW> describe scene (requires Azure)
<PW> detect objects (requires Azure)
<PW> who am i (Azure recommended but not required)
<PW> follow me (requires URG04 or Azure)

<PW> photo
<PW> batchfile ssssss
<PW> speak ssssss
<PW> find ssssss (requires Azure)
<PW> quit
<PW> shutdown
<PW> help

(where xx is an integer, ssssss is a string and <PW> is the API Password. All commands must be preceeded by the Password)

Turn Right Command

Instructs GotABot to turn right.

Example

COMMAND: "*turn right 45*"

RESPONSE: "OK > "

ACTION: Instructs GotABot to turn right 45 degrees.

Turn Left Command

Instructs GotABot to turn left.

Example

COMMAND: "*turn left 30*"

RESPONSE: "OK > "

ACTION: Instructs GotABot to turn left 30 degrees.

Forward Command

Instructs GotABot to move forward.

Example

COMMAND: "forward 17"

RESPONSE: "OK > "

ACTION: Instructs GotABot to move forward 17 inches.

Back Command

Instructs GotABot to move back.

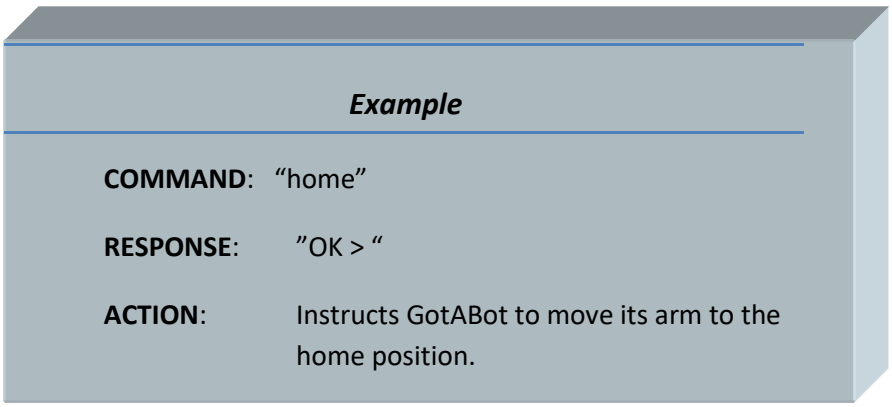
Example

COMMAND: "*back 6*"

RESPONSE: "OK > "

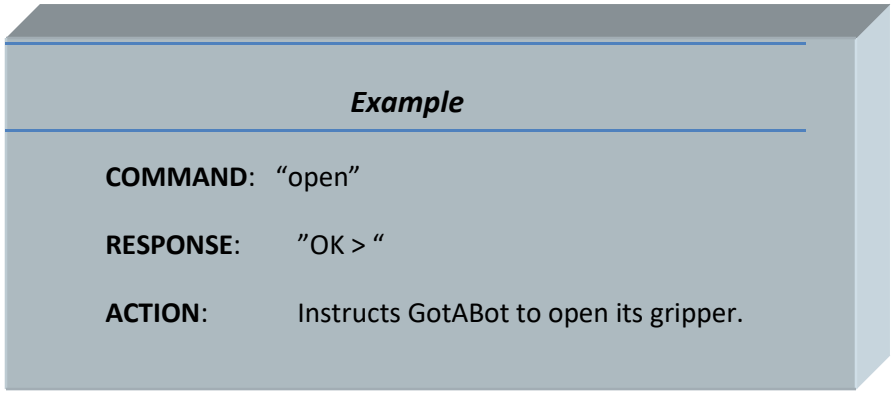
Home Command

Instructs GotABot to move its arm to the home position.



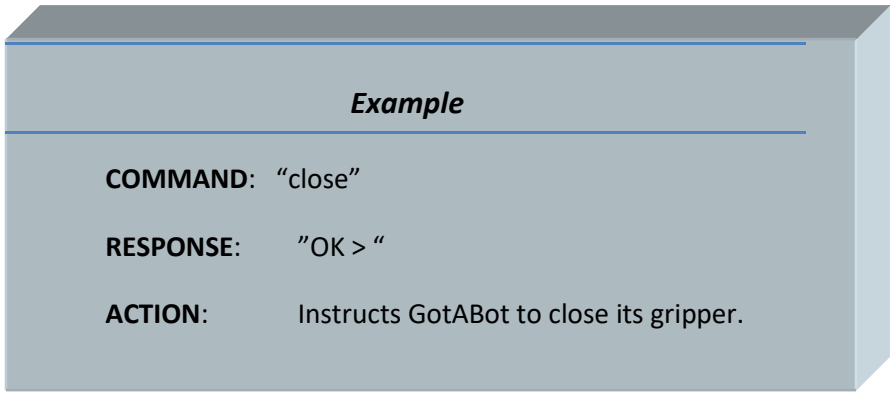
Open Command

Instructs GotABot to open its gripper.



Close Command

Instructs GotABot to close its gripper.



Pick Up Command

Instructs GotABot to drive to and pick up an object.

Example

COMMAND: "pick up 3"

RESPONSE: "OK > "

ACTION: Instructs GotABot to drive to Object 3, and pick it up.

Put Down Command

Instructs GotABot to drive to a location and put down whatever it is carrying.

Example

COMMAND: "put down 200"

RESPONSE: "OK > "

ACTION: Instructs GotABot put whatever it is carrying down and release it at a height of 200mm.

Set Gripper

Set the Gripper's position relative to the center of the Bot.

Example

COMMAND: "set gripper 110 200"

RESPONSE: "OK > "

ACTION: Move the gripper to 110mm from the Bot center and 200mm from the ground.

Locate Object

Locate an object. The Bot must be facing the object.

Example

COMMAND: "locate object 3"

RESPONSE: "OK > "

ACTION: Point the arm and camera to the object's presumed location and visually measure it's relative location. The results may be retrieved via the Get Status Full command.

Pick Up Command

Instructs GotABot to drive to and pick up an object.

Example

COMMAND: "pick up 3"

RESPONSE: "OK > "

ACTION: Instructs GotABot to drive to and pick up object 3.

Pull Command

Instructs GotABot to drive to and pull an object.

Example

COMMAND: "pull 3"

RESPONSE: "OK > "

ACTION: Instructs GotABot to drive to and pull object 3.

Arm Angles Command

Set the arm's joint angles.

Example

COMMAND: “arm angles 90 10 45 0 20”

RESPONSE: ”OK > “

ACTION: Set the arm's joint angles:
Shoulder = 90 degrees.
Elbow = 10 degrees.
Wrist Rotate = 45 degrees.
Wrist = 0 degrees.
Gripper = 20 degrees.

Set Destination Command

This command is equivalent to the GUI's Destination command.

Example

COMMAND: *"set destination 50 135"*

RESPONSE: "OK > "

ACTION: Instructs GotABot to plan a path from the Robot to map position 50 135, and then to issue driving directions to the Robot.

Set Pose Command

This command allows the calling program to set the robot's location, orientation and uncertainties. The command format is:

set pose <location x> <location y> <uncertainty x> <uncertainty y> <orientation> <orientation uncertainty>

Example

COMMAND: *"set pose 150 202 8 9 90 10"*

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot is located at location 150, 202 with an 8 inch uncertainty in the X axis, 9 inch uncertainty in the Y axis, and with an orientation of 90 +/- 10 degrees.

Set Orientation Command

This command allows the calling program to set the robot's orientation and uncertainty. The command format is:

set orientation <orientation> <orientation uncertainty>

NOTE: This command will only update the orientation when the robot is stopped and the orientation uncertainty is less than the existing uncertainty.

Example

COMMAND: *"set orientation 90 10"*

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot's orientation is 90 +/- 10 degrees.

Get Pose Command

This command will instruct GotABot to respond with the robot's current pose in the format:

<location x> <location y> <uncertainty x> <uncertainty y> <orientation> <orientation uncertainty>

Example

COMMAND: *"get pose"*

RESPONSE: "POSE = *150 202 8 9 90 10*
OK > "

ACTION: This instructs GotABot to report the current Robot pose.

Get Snack Command

Deliver a snack to a destination and then go to its Dock.

Snacks and other Objects may be defined by clicking *Configure->Objects*.

Example

COMMAND: *"get snack 1 55 89 90"*

RESPONSE: "OK > "

ACTION: This instructs GotABot to get Object No 1 and deliver it to location 55, 89 at an orientation of 90 degrees. Upon completion of delivery the Bot will go to its Dock.

Get Status Command

This command will instruct GotABot to respond with the robot's current status (*"docked" or "moving"* or "*stopped*")

Example

COMMAND: *"get status"*

RESPONSE: "stopped
OK > "

ACTION: This instructs GotABot to verbally report its status.

Get Status Full Command

This command will instruct GotABot to respond with the robot's current status

Example

COMMAND: *"get status full"*

RESPONSE: "Pose = 117, 166 +/- 12, 12 at 180 +/- 19 degrees
Pose = 186, 132 +/- 10, 10 at 187 +/- 10 degrees
Connected To Hardware = True
Battery Percentage = 97
Moving = False
URG Connected = True
Kinect Connected = False
Arm Connected = True
RCC Connected = True
Docking Mode = False
Get Object Mode = False
Driving To Mode = False
Guard Mode = False
Follow Mode = False
Pause Mode = False
Arm Moving = False
Shoulder Angle = 90.20898
Elbow Angle = -13.875
Wrist Angle = -0.5859375
Wrist Rotate Angle = -0.5859375
Gripper Angle = 2.34375
Obstacle Vector = 22.4722 at 13.85446 degrees.
Gripper Coordinates (Y,Z) = 244, 678
Object Coordinates (X,Y,Z) = 0, 610, 600
OK > "

ACTION: This instructs GotABot to report the full status - with no voice response.

Dock Command

This command is equivalent to the GUI's Dock command.

Example

COMMAND: *"dock"*

RESPONSE: "OK >"
ACTION: This instructs GotABot to Dock with its charging station

Find Command

Visually find an object using Azure.

Example

COMMAND: *"find person"*

RESPONSE: "71, 220, 167, 139
OK >"
ACTION: In the example above, a person was searche for and found in a rectangle defined by the top left corner at Pixel x=71, y=220 and width of 167 pixels and height of 139 pixels.

Get Path Command

This command will instruct GotABot to plan a path from the robot to a destination, and then to respond with a series of locations indicating the steps required to reach that destination. The command format is: get path <location x> <location y> (where location x and location y are the desired destination)

Example

COMMAND: "*get path 150 55*"

RESPONSE: " Path =
171 188
178 181
178 177
180 175
180 160
181 159
181 157
182 156
182 125
176 119
176 77
154 55
150 55
OK >> "

ACTION: Plan a path to location 150, 55 and then report the steps to reach that destination.

Localize Command

This command is equivalent to the GUI's Localize command.

Example

COMMAND: *"localize"*

RESPONSE: " POSE = *150 202 8 9 90 10*
OK > "

ACTION: This instructs GotABot to perform a localize operation (look for Land Marks or perform a range scan , deduce the Robot's position and orientation) and then to report the Robot's pose.

Stop Command

This command is equivalent to the GUI's Stop command. All active GotABot and robot activities should stop on receipt of this command.

Example

COMMAND: *"stop"*

RESPONSE: " OK > "

ACTION: Stop all movement and localization.

Guard Mode Command

Checks for motion by video analysis and for motion detected by a Scanner. Issues verbal challenges and takes photos when motion is detected.

Example

COMMAND: "*guard mode on*"

RESPONSE: " OK > "

ACTION: Enables guard mode. Similarly, "*guard mode off*" disables guard mode. Photos are stored in the *Guard* directory

Patrol Command

Periodically, go to a destination, pirouette while taking photos, and then return to the dock. Can be used in combination with Guard Mode for extra photo coverage.

Example

COMMAND: "*patrol 55 133 120*"

RESPONSE: " OK > "

ACTION: Every 120 minutes, GotABot will go to location 55,133 then pirouette then return to dock. Photos are stored in the *Photos* directory.

To exit Patrol Mode, send "*patrol exit*"

Get Object Command

Get an object, deliver it to a destination and then return to the dock. The object may either be in a container such as a cabinet or refridgerator, or the object might in an assessable location such as a shelf.

Objects may be defined by pressing *clicking->Objects*.

Example

COMMAND: "*get object 5 4 70 195 180*"

RESPONSE: " OK > "

ACTION: Open object 5, get object 4, and then deliver it to location 70,195 with an orientation of 180 degrees. Finally, return to the dock.

If the object is assessable without the need to open a container, use the following format:

*get object **999** 4 70 195 180*"

10. Objects

Generally, Objects are items which the Arm or a Vector Robot can pick up or manipulate. They are defined in the Objects Window (click *Configure->Objects*) where the name, location, offsets, etc are entered.

Instructions for picking up or manipulating these objects are controlled by either the API or indirectly via voice commands.

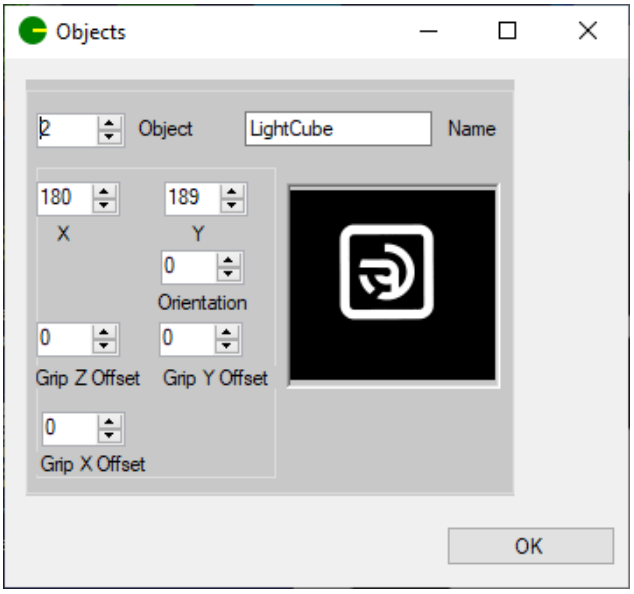
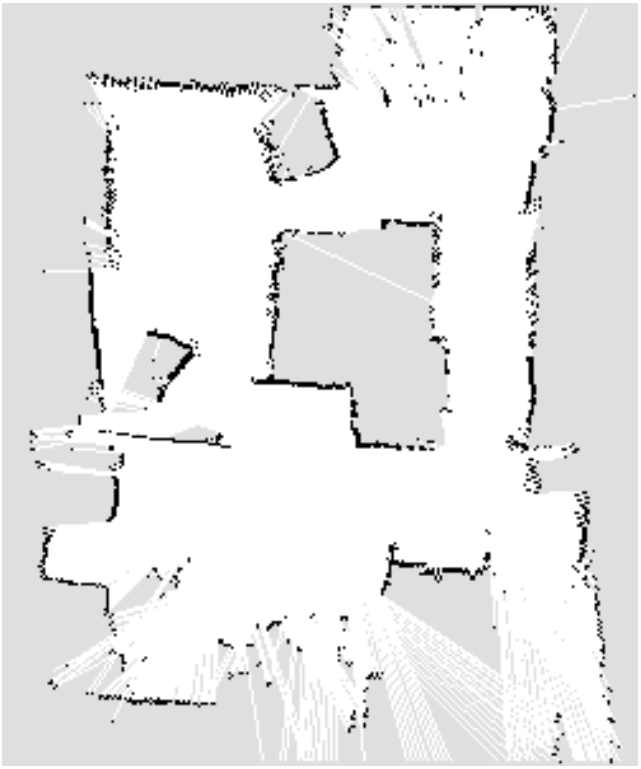


Figure 8 Objects Window

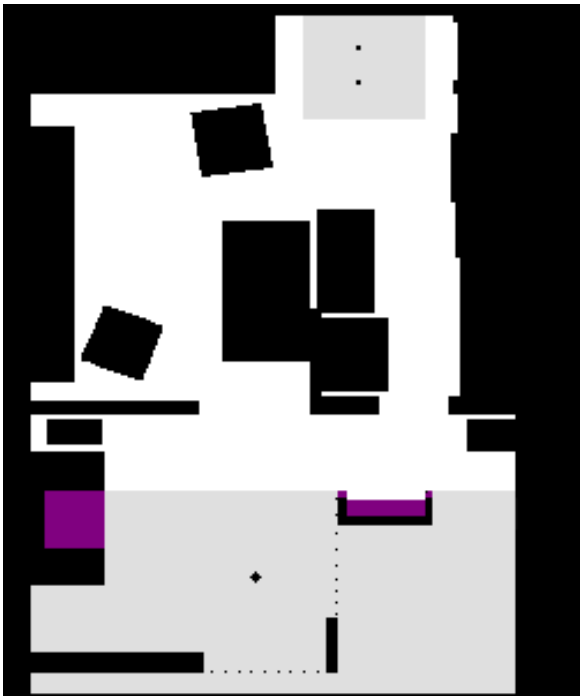
11. SLAM

SLAM, or Simultaneous Localization And Mapping, has long been one of the holy grails of the robot navigation world. After all, making a map by hand can be a fussy and aggravating exercise of measurement and transcription. Far better to let our clever robots handle these mundane details.

Over the years, many schemes utilizing various sophisticated algorithms and differing sensors have been developed and offered. Indeed, many academic careers have been built on this research.



9 Kinect Derived SLAM Map



10 Hand Drawn Map

11 URG04 Derived SLAM Map

At times, SLAM has been declared a 'solved problem' but new information on inadequacies of existing methods and new methods of addressing these inadequacies continue to be developed.

GotABot uses a variation of "Particle Filtering" for SLAM operations.

SLAM operation is started by clicking *Tools->SLAM*.

The Trouble With SLAM

In a word, the trouble with SLAM can be summed up as 'accuracy'. Unless the Robot's odometry or localization technique is exceptional and the ranging devices measure flawlessly, the resulting map will be less accurate, likely considerably less accurate, than a map generated by the user. For example, compare the user generated map with its equivalent Kinect derived SLAM map. Not as good, eh? Fortunately, even given the lower accuracy, the SLAM map is still adequate for most navigational purposes.

Kinect vs URG04 Map

As expected, the URG04 with its much wider FOV produces a map far superior to the Kinect's. Wall edges are more true with less noise and generally closer to their actual locations. More important than this is the ease of use in creating the map. The time required to map with the Kinect was several times longer and required judicious application of pirouettes and careful movements to keep the Particles from diverging and failing. By contrast, mapping with the URG04 was relatively easy, requiring only directing the Bot around the area to be mapped.

SLAM Requirements

The most obvious requirement for SLAM operation is, of course, to have some method of determining empty from occupied space. GotABot may employ a Kinect or a URG04 to achieve this.

SLAM Operation

To initiate and control the map making process, follow these steps:

STEP 1) Open the SLAM form by clicking *Tools->SLAM.*

STEP 2) Select a SLAM map when prompted. An unexplored SLAM map is a uniform grey (Red=223, Green=223, Blue=223). An example map is included in the installation package.

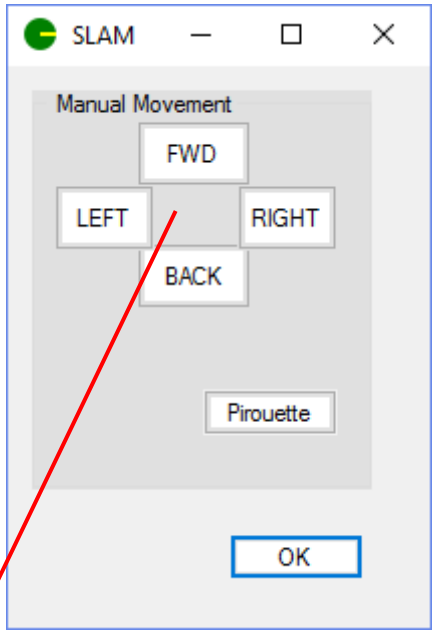


Figure 12 SLAM FORM

STEP 3) Use the Robot Pose form to set the map pose to match the actual robot's pose.

STEP 5) Use the *Manual Movement* controls to move and map.

STEP 6) Repeat step 5 until the map is adequately explored or the battery is drained.

STEP 7) Save the map. (Click *Configure->Map->Save Map).*

STEP 8) (Optional) The map may be enhanced or corrected by editing in a graphics editor such as MSPaint.

TIP: Slow Down

Unless you are rocking an exceptionally fast laptop, keep all Bot velocities and accelerations low.

12. AZURE

GotABot has an optional interface to Microsoft Azure Cloud Services. These services are used for enhanced Facial Recognition as well as object recognition.

1) Azure is enabled via the Video Form (On the main GotABot window click *Tools->Video*. When the Video Window / Form opens click *Image Recognition->Azure Enable*.)

2) Sign up for an Azure free account

3) Sign in and enter the account portal.

4) Create two resources: Under the *AI + Machine Learning* category - subscribe to both Computer Vision and to Face.

5) Copy and paste the resulting Key and EndPoint of each of the resources into GotABot. (In the Video Window click *Image Recognition* and click the corresponding menu item for each of these four items.)

6) Job Done!

APPENDIX A: Evolution Robotics ER1

RCC Configuration

In the RCC application, Click *Settings->Remote Control*.
Click "*Allow API control of this instance*"
Port = 9000
Password = blank (i.e. no password)

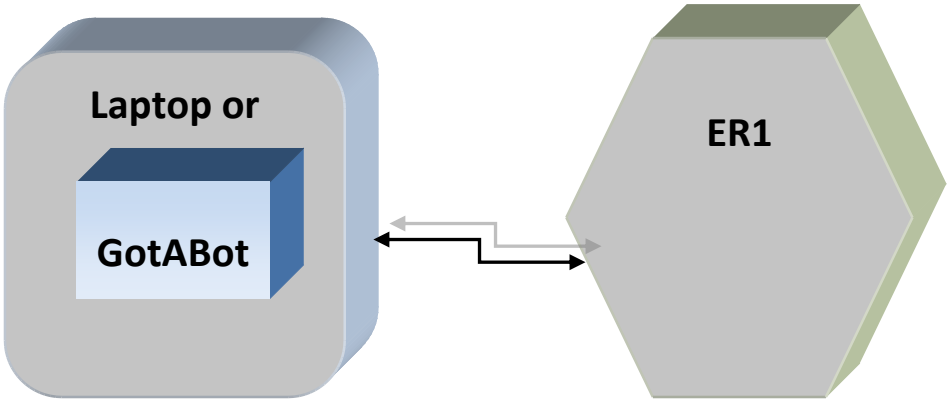
ER1 Driver Installation For Windows 10

The original hardware drivers do not work with any Microsoft Windows version post XP. To fix this, the GotABot download package contains new drivers which operate with more modern Windows.

NOTE. This driver installation uses modified FTDI VCP v2.12.12 drivers. They were modified by FTDI's FT_INF utility so as to work with the ER1's obsolete interface. This makes them unsigned drivers requiring a non standard installation.

To setup / use:

a. Enter the unsigned driver installation mode:
   i. Open the 'Run dialog' (*press Win+R*)
   ii. type `shutdown.exe /r /o /f /t 00`
   iii. After rebooting into the *'Options Menu*', click *'Troubleshoot'* and then *'Advanced Options*'.
   iv. Click *'startup settings*' and then '*restart*'.
   v. After rebooting, click option *7 'Disable driver signature enforcement*' and then reboot.
b. Navigate to the RCM Driver folder of the download package.
c. Right click on the file *ftdibus.inf* and click on *install*. Follow the prompts to install.
d. Right click on the file *ftdiport.inf* and click on *install*. Follow the prompts to install.
e. Connect the ER1. to your laptop.
f. Start GotABot
g. Select ER1.
h. Save settings and restart GotABot.



Optional ER1 Hardware

The ER1 has the capability of accessing both digital and analog ports through its Robot Control Module (RCM). GotABot exploits this capability by employing them to access various sensors and devices which extend and enhance its basic operation.

Use of these capabilities is optional and not required for basic operation. Robot's other than the ER1 may enjoy similar capabilities through use of the appropriate API command. ER1 bumper switches are enabled by clicking *Configure->Options->Bumper Switches*.

ER1 Bumper Switches

Gotabot supports three Bumper switches; left, right and rear. Behavior due to a Bumper Switch activation is the same as per the Bumper Switch API command.

Bumper Switch Wiring

RIGHT SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DIO	21

LEFT SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DI1	21

REAR SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DI2	22

TIP: Get Some Bumper Switches

Bumper Switches are such simple devices that their utility is frequently under appreciated. I would argue that given the inherent contrariness of a typical robot, bumper switches are well nigh indispensible.

APPENDIX B: Random Hints And Tips

Accessing a laptop while it is tethered to a robot is inconvenient. I use RealVNC (https://www.realvnc.com/) to login from my desktop. Much easier.

If several programs , for instance GotABot and RCC, need access to the same webcam install a program such as ManyCam to enable multiple feeds from a single webcam.

If GotABot is installed in a Windows XP system then make sure that the system has been updated with Net Framework.

If the robot 'fishtails' when coming to a stop, increase the 'Power Stopped' (press *Configure->Bot->Power Stopped*).

A camera or a Kinect will see more close in details if it is mounted near the rear of the robot.

For faster localization, use the smallest map which still shows the area which the robot can observe and operates within.

Recommended Speeds for the ER1:

Acceleration:	60 cm/s^2
Velocity:	20 cm/s
Angular Acceleration:	90 $Degrees/s^2$
Angular Velocity:	40 Degrees/s

Example Power Settings for the ER1:

Power Moving:	65
Power Stopped:	6

APPENDIX C: Heathkit Hero-1

The Heathkit Hero-1 is one of the all time great hobbyist robots. First produced in 1982, it has all the features required by GotABot; mobility, odometry, range sensing and an interface. Unfortunately, its early technology results in limited sensor range and discrimination, poor steering accuracy, and glacially slow processing. Still, extending this iconic platform's capabilities with a modern laptop or tablet is educational and a hoot to boot.

Figure 13 Heathkit Hero-1

Hero-1 / GotABot Configuration

Follow these steps:

1) The Hero must have a Serial or USB interface as well as a the requisite 1.U or 1.3 ROM with 4 MHz crystal. Configure the interface for 9600 baud.
2) Create a map.
3) Connect the laptop or tablet to the Hero. (See example).
4) Turn on the laptop and Hero.
5) Initialize Hero by pressing *RESET* then *3* then *1* on the keypad
6) Start GotABot.
7) Click *Configure->Bot->Bot Select* and select *Heathkit Hero.*
8) Click *Configure->Bot->Bot Com Port* and select the port connected to the Hero.
9 Click *File->Save Settings* and then restart GotAbot.
10*)* Follow the onscreen instructions to download an assembly program to the Hero.
11) Test your setup by Localizing and Manual movement control.
12) Job Done!

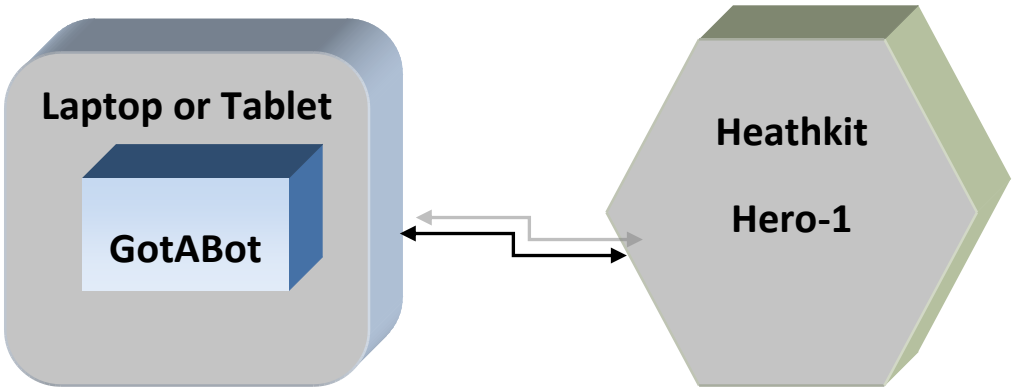


Figure 14 Hero-1 / GotABot Configuration

Hero To Tablet Connection Example

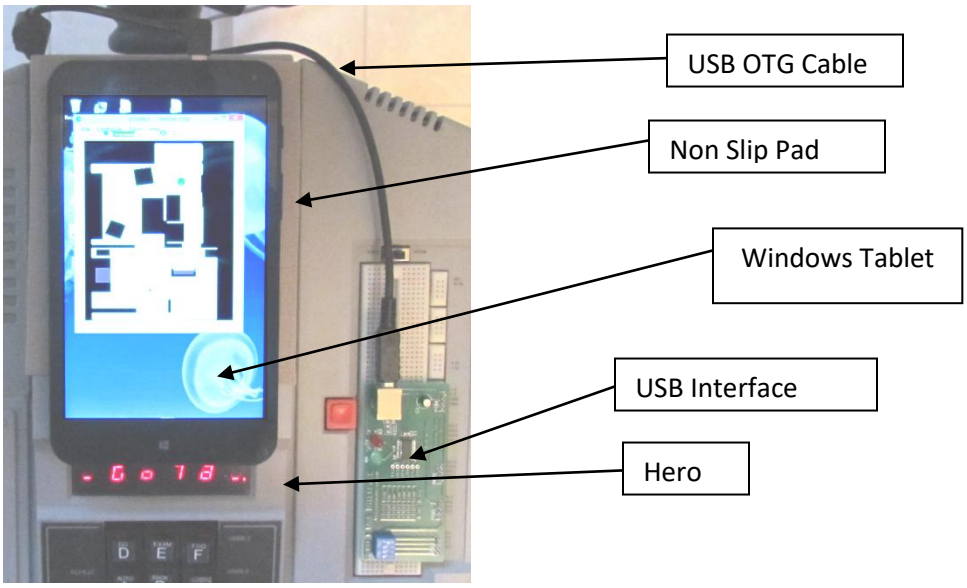


Figure 15 Hero To GotABot USB Connection

Laptop Remote

Operating the laptop while it is being carried hither and yon is challenging and inconvenient. I use VNC to log into the laptop from my desktop.

Hero-1 Hardware Compensation

If your Hero's hardware is not perfectly aligned then there are some settings which might compensate.

Hero-1 Steering Compensation

If your Hero veers left or right instead of going dead ahead, then click *Configure->Bot->Steering Bias Correction* and enter an adjustment.

Hero-1 Head Offset Compensation

If your Hero's head does not point straight ahead then click *Configure->Scanner* and adjust the *Scanner Offset in Degrees* field to compensate.

Hero-1 Distance Compensation

If your Hero does not accurately travel distances commanded then click *Configure->Bot->Distance Compensation* and enter the correction factor.

TIP: Keep it simple

As discussed, the Hero-1 has limited sensing and movement accuracy. Successful operation will likely require an environment with extra wide pathways and uncluttered walls which return good echoes for the ultrasonic ranger.

APPENDIX E: Arm

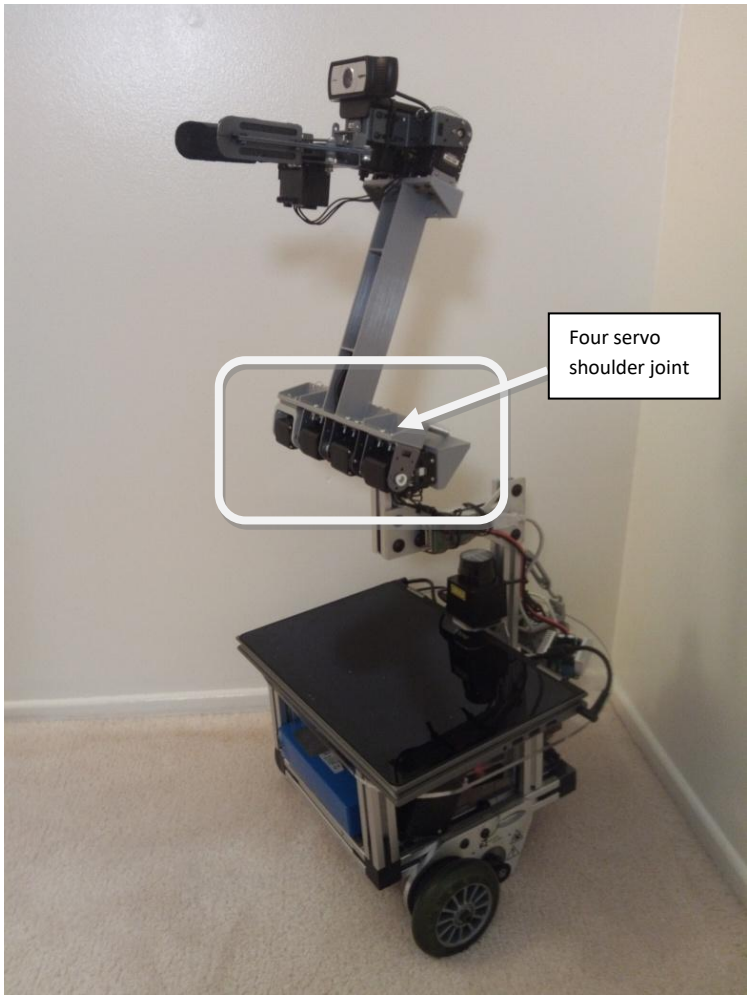
GotABot's arm is a CrustCrawler AX-12AUSB with the rotating base removed and mounted as shown. A Robotis USB2Dynamixel is used to interface between the arm and laptop.

NOTE: The Wrist Rotate servo is mounted upside down (vs the original build instructions)

Servo	Servo Address	Direction
Shoulder Right	2	Reverse
Shoulder Left	3	Forward
Elbow Left	4	Forward
Elbow Right	5	Reverse
Wrist Rotate	6	Forward
Gripper	7	Forward

Four Servo Shoulder Joint Option

While the original CrustCrawler arm worked well, it was shorter than desired. Simply lengthening the arm would require more shoulder torque than the two existing Dynamixel AX-12A servos could supply. One option would be to upgrade the shoulder servos with stronger ones. This is a relatively simple swap only costing money. So much money. More than was feasible. Luckily, two additional AX-12A servos were available and so a slightly Rube Goldbergish 4 servo shoulder joint was constructed and which worked surprisingly well.



Servo	Servo Address	Direction
Shoulder Right2	1	Forward
Shoulder Right	2	Reverse
Shoulder Left	3	Forward
Elbow Left	4	Forward
Elbow Right	5	Reverse
Wrist Rotate	6	Forward
Gripper	7	Forward
Shoulder Left2	8	Reverse

Figure 16 Four Servo Shoulder Joint

To enable the 4 Servo option open the GotAnArm window by clicking *Tools-> Arm* and then clicking the *'4 Servo Shoulder*' radio button in the GotAnArm window. Update the various *'Arm Lengths*' and other fields to reflect the arm's dimensions and flex compensations.

The STL files for the lengthened Shoulder to Elbow arm segment as well as the Base mounting component are included in the downloaded files.

Spring Assist

Whether using the regular or optional four servo shoulder joint, the arm's lifting capabilities may be significantly increased by adding springs to the shoulder and elbow joints. Figure 16 shows a simple method of mounting them. Remember to update the *'Flex Compensation*' fields.

Figure 17 Arm (Rear View) Showing Springs

APPENDIX F: GBot Controller

So, your RCM is not working and your ER1 is useless. Perhaps old age, random chance or a poorly conceived experiment has broken it. Happens. GBot is a replacement which works with the ER1 hardware (but not the original RCC software. No matter - it works with GotABot and, as a bonus, draws less power.)

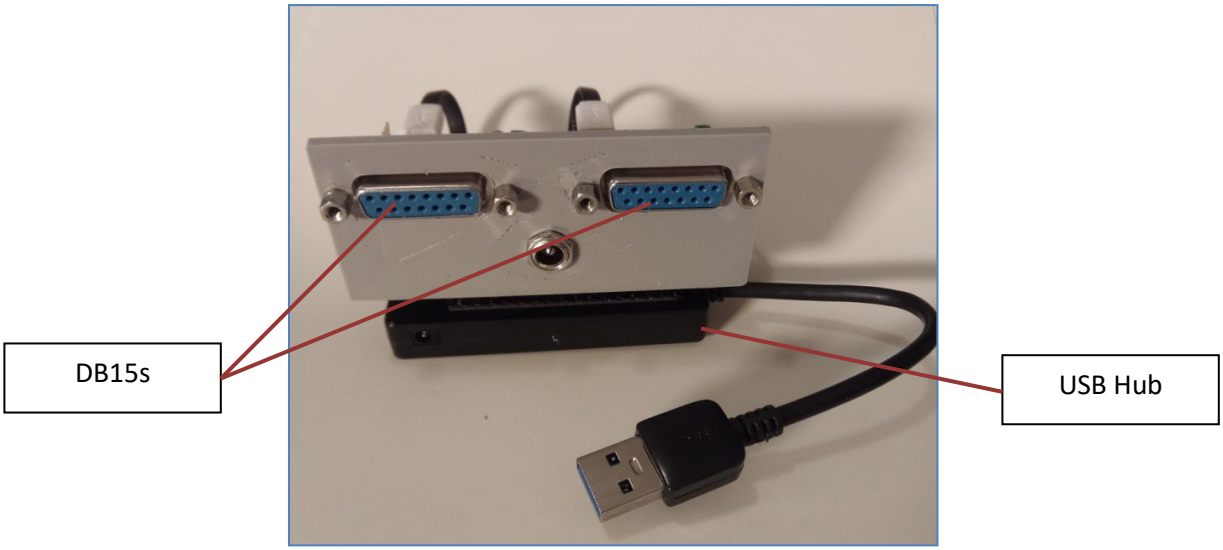


Figure 18 GBot Controller Front

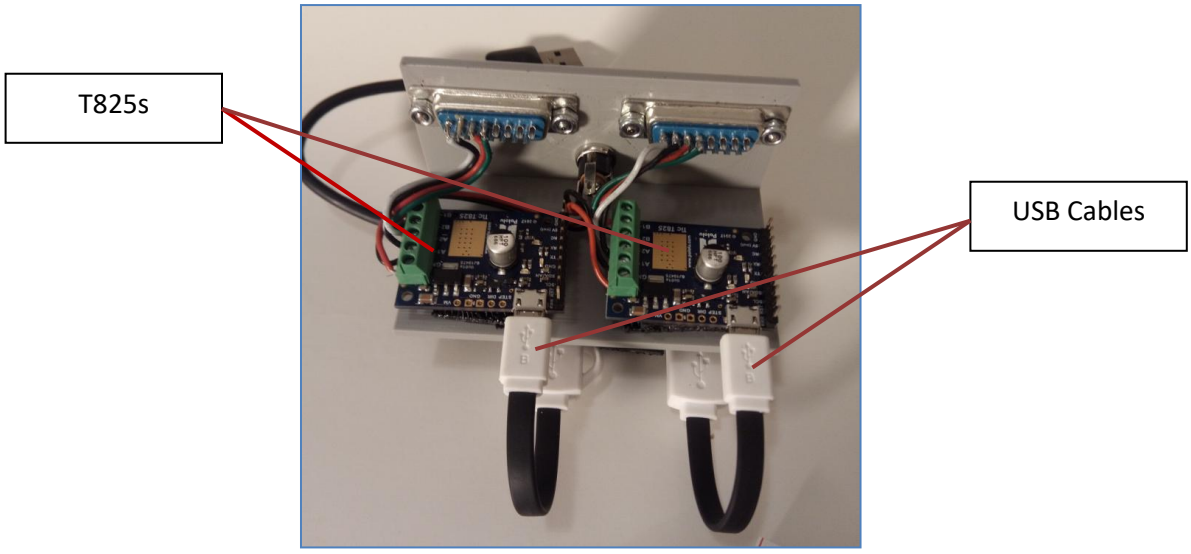


Figure 19 GBot Controller Rear

Qty	PARTS LIST	US$
2	Pololu T825 Stepper Controller	60.00
2	DB15 Female Panel Mount	4.00
2	USB cables USB A to micro-B	7.00
1	Power Jack	2.00
1	USB Hub	10.00
1	Enclosure (3D Printed or original RCM Enclosure)	
TOTAL		**83.00**

Table i GBot Parts List

NOTE: A 3D printable stl file for a partial enclosure is included in the download package

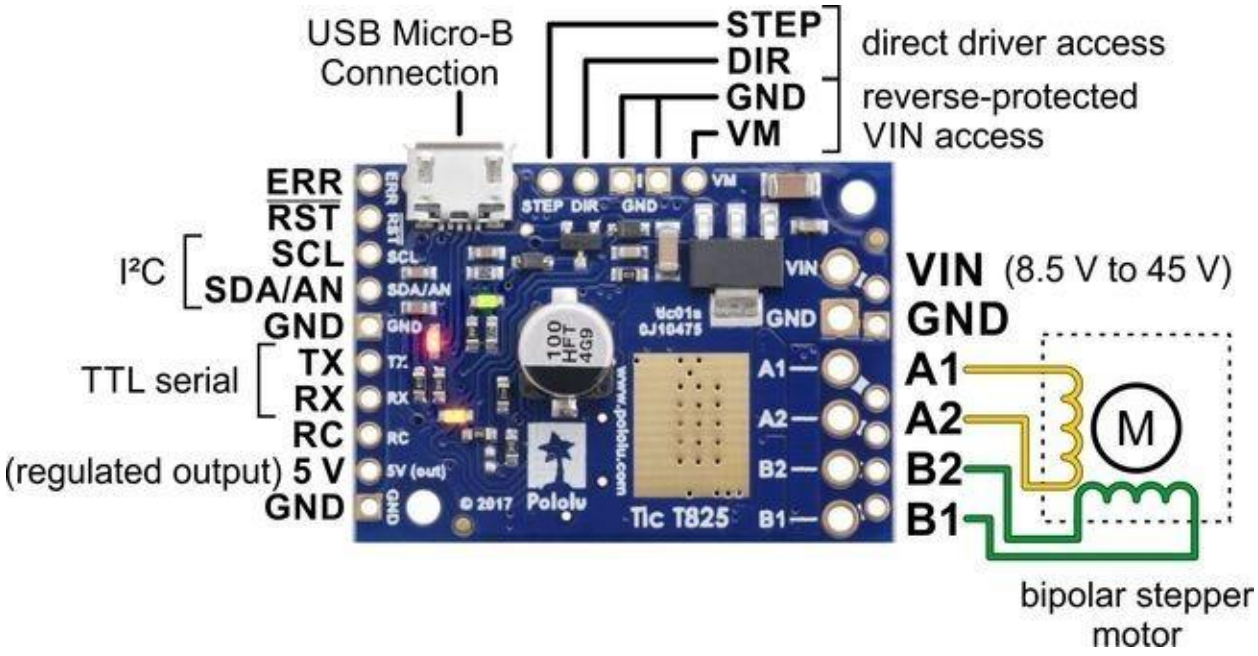


Figure 20 Pololu T825

GBot Wiring Table Pololu T825	
A1	DB15 - pin 2
A2	DB15 - pin 1
B2	DB15 - pin 4
B1	DB15 - pin 3
Vin	Power Jack - +12V
GND	Power Jack - GND

Table ii GBot Controller Wiring

NOTE: Not shown here but a small fan should be added for cooling the Pololu T825. Recommended if "*Power Moving*" is set to more than about 50% or if the Bot Speeds / Acceleration are set high.

Setup

1) Download the T825 Software and Drivers from Pololu.
2) Install and select the Add To Path option.
3) Connect the T825s to power and the PC. (see Table iii GBot Controller Wiring and Figure 21 GBot Controller Rear).

4) Open the ticgui.exe utility. For each of the T825s (select via the 'Connected to' box):
   a) Select the 'Input and motor settings' tab
   b) Unselect the 'Enable command timeout' box
   c) Set the Baud rate to 115385
   d) Click 'Apply settings'
5) Using the ticgui.exe utility, verify that GBot can control the wheel servos.
6) The POLOLU software is not required during GotABot operation.
7) In the GotABot main window:
   Click *Configure->Bot Select->GBot*.
   Also under the *Configure* menu, update the *Bot Diameter* and *Wheel Separation* distances.
   Click *Configure->Options* and ensure that *Simulated Movement* is not checked.
   Then click *File->Save Settings*

APPENDIX G: Anki Vector

Setup

1) If not done already, use the Anki Vector app to connect Vector to wifi and update his files.
2) Generate Vector's configuration file by installing and using Vector Explorer
3) GotABot configuration:
   i. Click *Configure->Bot->Bot Select* and choose *Anki Vector*
   ii. If you have more than one Vector enter its Serial Number (Click *Configure->Bot->Serial Number).* Your Vector's Serial Number may be found on its bottom (ex. 00e20100). If you have only one Vector then you may leave the Serial Number setting as *xxx*.

Key Control

As well as using the normal Menu selections, Vector may be controlled by the following keys:

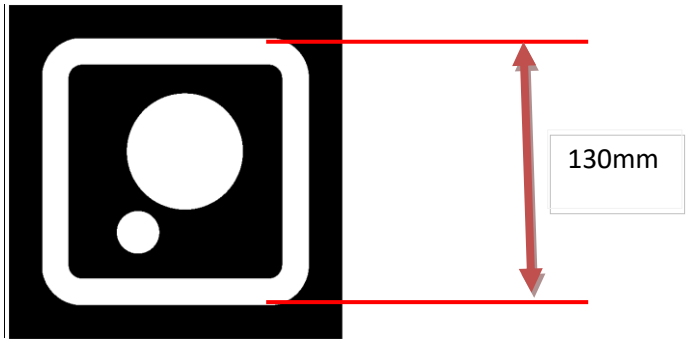
Arrow keys	Drive Forward, Left, Back and Right
W,A,S,D	Drive Forward, Left, Back and Right
U	Raise Head
J	Lower Head
T	Raise Arm
G	Lower Arm
<Enter>	Speak Text

Localization

Localization is performed via LandMarks. However, there are some important differences from the standard method:

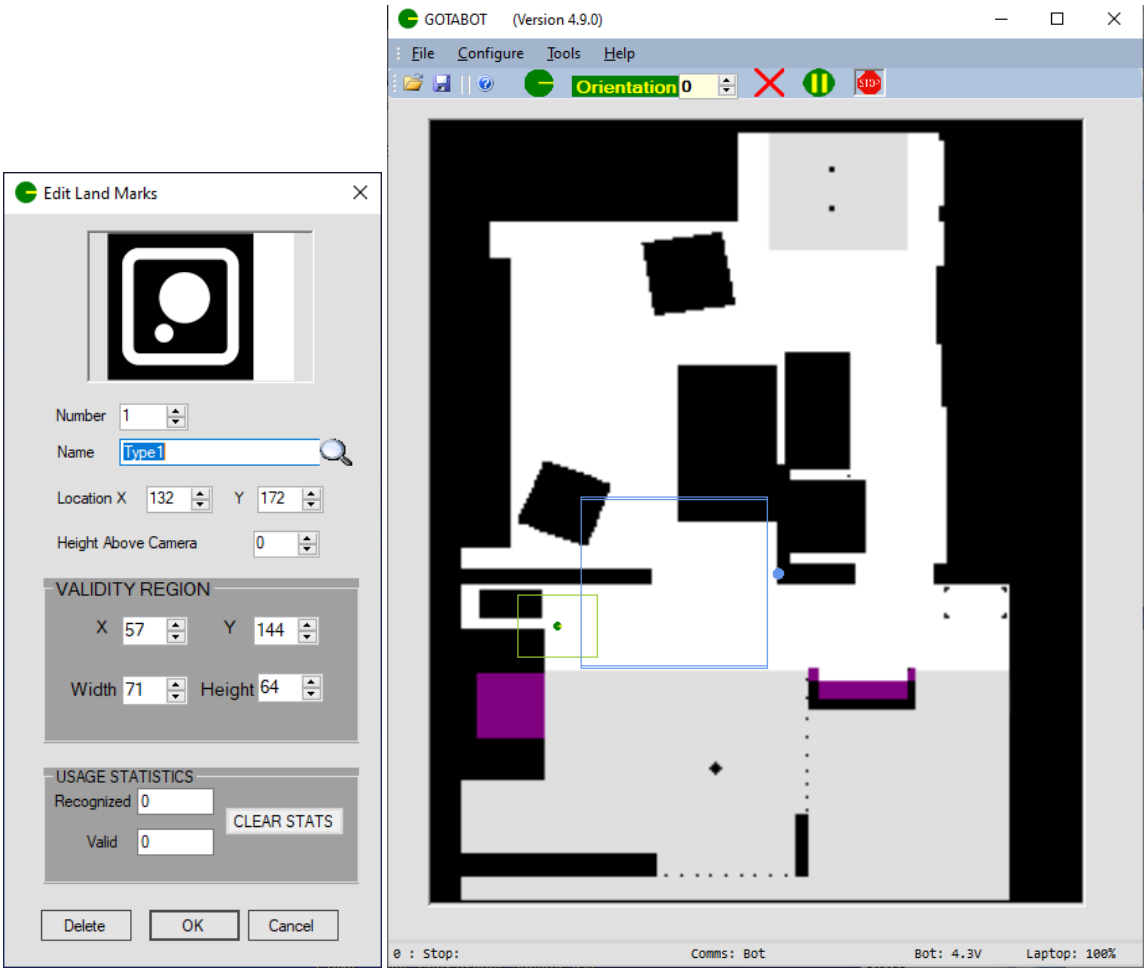
1. Visual processing is performed on Vector itself - no need for the RCM application.
2. There are eighteen different LandMarks:
   i) *Charger*
   ii) *LightCube*
   iii) *Type1*
   iv) *Type2*
   v) ...
   xviii) *Type16*

   Pictures of *Type1* through *Type16* LandMarks are included in the '*Vector Objects.pdf*' file which is located in the *AnkiVector* directory of the download package. The outside edges of Type1 through 13's white squares surrounding the central symbol are approximately 130mm per side.`



For applications requiring smaller landmarks, Types 14 through 16 are approximately 40mm per side.

4. Enter each LandMark's Location and Validity Region using the Edit Land Marks window.



5. The LandMarks must be placed near the floor, at Vector level, for Vector to recognize them.
6. Multiple LandMarks of the same name may be used, but try to keep them physically separated so that there is minimal confusion about which one is observed.

Other

Configure the Dock (Charger) for location, orientation, and trigger voltages.

Vector Screen Display

Vector can display many interesting things on its screen; Webcams, YouTube, IP Cameras, Games, and more.

Webcams

1) Open the Video window (In the main GotABot window click *Tools->Video*).
2) In the Video window click *Screen* and select the webcam you wish displayed.

YouTube

1) This will require the use of an additional program, ManyCam . This program is free to use with some restrictions on features. Start ManyCam and select YouTube as the source, and paste in the video's URL.
2) If not already open, open GotABot's Video Window (In the main GotABot window click *Tools->Video*).
3) In the Video window click *Screen* and select the ManyCam Virtual Webcam.

IP Camera

1) This will require the use of an additional program, ManyCam . IP Camera use requires a subscription to ManyCam. Start ManyCam and select *IP Cameras* as the source, and paste in the video's *URL*.
2) If not already open, open GotABot's Video Window (In the main GotABot window click *Tools->Video*).
3) In the Video window click *Screen* and select the ManyCam Virtual Webcam.

Desktop Window

1) This will require the use of an additional program, ManyCam . Start ManyCam and select *Desktop and then the desired Window* as the source.
2) If not already open, open GotABot's Video Window (In the main GotABot window click *Tools->Video*).
3) In the Video window click *Screen* and select the ManyCam Virtual Webcam.

Area Under Cursor

1. Very useful for displaying a remote users face during Virtual Presence operation.
2. This will require the use of an additional program, ManyCam . Start ManyCam and select *Desktop->Area Under Cursor* as the source.
3. If not already open, open GotABot's Video Window (In the main GotABot window click *Tools->Video*).
4. In the Video window click *Screen* and select the ManyCam Virtual Webcam.

APPENDIX H: Facial Recognition

There are actually two Facial Recognition systems - the regular standard OpenCV type and the more accurate AZURE recognition for which you need a subscription from Microsoft (free).

Let's start with the simplest -OpenCV using your photo library for face training:

Open the Video window - click *Tools->Video*

In the Video Window click *Camera* and select your webcam and resolution. Higher resolutions allow recognition at greater distances but are, of course, slower. Don't worry about FOV or offset fields for now.

Click *Image Recognition->Enable Face Recognition*

Click *File->Save Settings*

Click Image *Recognition->Train Photo.* Enter the person's name and select a photo of that person's face. The OpenCV facial recognition system requires multiple pictures of each person for accurate results so repeat the Photo Training many times (at least 10?). Choose photos taken from different angles, lighting conditions, and facial expressions.

(Alternatively, instead of using your photo library, you may use a webcam to capture faces by clicking *Train Webcam*. Look into the webcam and tilt your head at different angles as a series of 10 portraits are taken)

You can check and edit the training results in the *TrainedFaces* directory.

APPENDIX I: Kinect Notes

1) GotABot can use an optional *Kinect Sensor For XBOX 360*.

2) The Kinect requires a USB/AC interface adapter so that it can connect to a laptop. Buy one on eBay. Remove the AC adapter and wire the USB interface to your 12V battery. Be careful with polarity.

3) Download the Kinect v1.8 software from http://www.microsoft.com/en-us/download/. Install the SDK. The Toolkit is optional but darn interesting.

4) Power up and in the Device manager you will see that four new devices have been installed:
   Kinect for Windows Audio Array Control,
   Kinect for Windows Camera,
   Kinect for Windows Device,
   Kinect for Windows Security Control.

5) GotABot will automatically detect and connect to the Kinect.

Kinect Limitations and Workarounds

Compared to a typical laser scanner, a Kinect is limited by the following:

i) a narrow FOV which reduces the chances of a good localization

ii) a greater minimum distance for object detection which reduces the chances of detecting obstacles and also reduces the chances of a good localization

There are a number of measures which may be used to alleviate these limitations:

i) In areas where localization are difficult, add a feature to the room and map. A cunningly placed object may provide enough additional information for good localization. A potted plant against a featureless wall, or a small cabinet, bookcase, or other object might make all the difference required.

ii) Alternatively, a splash of Purple (Red=128, Green=1, Blue=128) on the map will keep your robot out of areas where localization is problematic.

iii) Increasing the *Bot Diameter* setting will keep your Bot further from walls and objects, thereby lessening the risk of collisions. Note that this might limit the ability to route through narrow passage ways and it will increase the passage time through narrow openings.

iv) Consider using LandMarks in addition to the Kinect. GotABot will look for your well chosen LandMarks when it can't localize with the Kinect.

v) Place the Kinect as far to the rear of your Bot as possible. This not only slightly increases the area scanned, but also decreases the object detection minimum distance from the front of your Bot.

APPENDIX I: Scripting

For Voice or API control it is convenient to have a series of actions performed with only a single command. This is implemented by issuing an API command of *'batchfile* `ssssss'or a Voice` command of *'trigger batch file* `ssssss'` (where ssssss is replaced with the name of the batch file). The batch file in turn calls a program or script (python, perl, vb, c++, whatever) which issues a series of API commands .

The robot's status / progress may be monitored by means of the 'get status full' command

APPENDIX J: Drawing

GotABot can command an Anki Vector to draw any of a variety of simple images. Sample image files may be found in the 'Drawing' directory. These images are written in a simplified form of GCode, and additional images may be created. GCode commands available are G0 (drive to location), G1 (draw a straight line), G2 (draw a clockwise curve) and G3 (draw an anti-clockwise curve). An excellent and easy to understand explanation of these commands may be found at https://makezine.com/2016/10/24/get-to-know-your-cnc-how-to-read-g-code/.

To draw an image:

1) Attach a pen or marker to Vector's lifting arm (see https://www.thingiverse.com/thing:4544386/files).
2) Place Vector at the top left of a sheet of paper (paper in portrait mode). Orient Vector to face towards the bottom left.
3) click *Tools->Draw->Draw File* and select the gcode file you wish to draw.

NOTES:

1) The Pen Tip to Bot center distance is critical for drawing accuracy. Measure this distance and enter it by clicking *Tools->Draw->Pen To Bot Center (mm).*

2) Curve drawing (G2 and G3 commands) is not terribly accurate. This can be somewhat improved by adjusting the end point of the G2 or G3 command and by changing the Curve Compensation field (click *Tools->Draw->Curve Compensation*)

APPENDIX K: Games

Pong

Play Pong against an Anki Vector! Press *Tools->Games->Pong* to start. Vector controls the left paddle. You control the right paddle by placing your hand near or far from Vectors front. First to 11 points wins. Restart or new game by touching (not clicking) Vector's back buttons.

APPENDIX L: Virtual Presence

Combine a Zoom Conference Call with a robot and you have a low cost Virtual Presence system. For our purposes, virtual presence is defined as the ability to operate a robot remotely while receiving video and audio feeds. Depending on the robot, remote manipulation of a robotic arm may also be supported.

Virtual Presence.

Imagine being able to physically interact with a far off environment.
A grandparent playing with a grandchild.
A parent touring an offspring's first apartment.
A traveler and a beloved pet.

All possible by combining the popular Zoom Conferencing app with a robot.

Zoom Conference Calls

The Zoom video conference call application is well suited for supporting virtual presence as it provides not only screen sharing but remote control options. In the following, the Host is the owner of the robot and initiates the Zoom call. The participant(s) are the remote users who wish to control the robot.

1) Security: Since this will involve giving limited control of GotABot to another, some attention needs to be given to minimizing risk to your computer and robot:
    1. Only give control to someone you trust. Instruct everyone to be careful and not drive the robot off a table or down any stairs.
    2. Although GotABot's Video window should not allow access to your documents and settings it is good practice to set up a Window's guest account and run Zoom and GotABot from there.
2) The Host begins by starting a Zoom video conference call.
3) The Host then shares GotABot's Video window.
4) Conference participants then request remote control of GotABot. The Host grants control.
5) The robot is controlled via the Manual Control keys:

Arrow keys	*Drive Forward, Left, Right and Back*
U,J	*Raise, Lower Head*
T,G	*Raise, Lower Arm*
F,H	*Arm In, Out*
<Enter>	*Speak Text*

GLOSSARY

API	Application Program Interface.
Bot	Short for Robot.
ER1	Robot manufactured by Evolution Robotics.
FOV	Field Of View. The diagonal angle, in degrees, captured by the camera.
IFTTT	If This Then That (Web application for voice control via Google Home or Alexa)
Land Mark	User selected images that the robot can recognize and measure range and direction to.
Localization	Determination of a Robot's pose through the use of Land Mark recognition or other sensory input.
Pose	The robot's position in space, quantified in the x, y and theta dimensions.
RCC	Robot Control Center. Software for image recognition.
RCM	Robot Control Module. Hardware component of the ER1.
Recognizer	A program which uses the robot's camera and performs object recognition on the image . This program must be able to determine the range to the object and relative angle. The Recognizer is integral to the Landmark recognition process.
SLAM	Simultaneous Localization And Mapping
USB OTG	On The Go. Enables devices such as some Tablets to act as USB hosts.
VGA	The VGA or Video Graphics Array standard specifies a resolution of 640 by 480 pixels.
YMMV	Your Mileage Might Vary.