MarcoPoloBot

1. About the Challenge ............................................................................................................................. 2
2. Searching for Information ...................................................................................................................... 3
3. Installing the Software ........................................................................................................................... 4
4. Learning to use basic/built in features with simulated robot ................................................................. 4
5. Thinking/Designing MarcoPoloBot system on paper ........................................................................... 5
6. Implementing the Design, Testing MarcoPoloBot on the Real Robot .................................................. 5
8. Additional Functionalities’ (open to your creativity) ............................................................................ 6

Vishwanathan Mohan

(Vishwanathan.mohan@essex.ac.uk)
1. The Challenge

The challenge is inspired by the Venetian traveler Marco Polo. The challenge for you is to create a MarcoPoloBot instead!

A one line description of the MarcoPoloBot challenge is as follows

“Design a simple system that will make the NAO robot walk to specific continents in a world map, when commanded by speech”

As an example, if the user says “Europe”, the robot must walk to Europe (in the world map), and so on. The basic challenge can be further improvised by adding other features (based on your own creativity and imagination...).

While working on this challenge, you will gain the following skills (and hopefully be curious to learn more).

- Finding information from the web about the Nao Robot, its basic hardware
- Installing a easy to use robot simulation environment called Choregraphe
- Learn to use basic built in features available in Choregraphe, for example to make the robot move to some location, recognize words
- Thinking about what functionalities you will need to create a MarcoPoloBot
- Design a full “MarcoPoloBot” system using/connecting building blocks appropriately
- Testing your design on the real robot
- Thinking about additional features you would like to have (for example, making the robot more interactive, may be make the robot say something about the continent where it travels)
- Working as a team, Allocating tasks to team members, Monitoring progress, Taking actions if things don’t work as expected
- Presenting your work to an audience
- Learning to take a “very general instruction- like design MarcoPoloBot” and come up with a detailed design, implementation and test. In general too, the ability of a good graduate is to work out the details from a general specification.

The hope is that some of these skills are quite general and may apply to any other task/project you undertake in the future. Of course don’t worry if you get stuck- you can always speak to your team mentor (Dr. Vishuu Mohan) for suggestions.

1.1 General information

For this challenge you will be working in groups of 6. You will need to come up with the design, discussing and working together as a team. Also you will need to allocate tasks to team members (who will do what, when, how), monitor progress, take actions if something does not work as expected and so on....

In general, every team member is expected to put around 20 hours of work.

And don’t forget to shoot videos, take pictures (both for your presentation and the 10 seconds a day video)
The robot simulation environment Choregraphe can be freely downloaded (see section 3). It’s also installed in all the Lab computers. All teams will be given access to the NAO robot to test your design.

2. Searching for Information related to your Challenge
To design the MarcoPoloBot system, you might want to learn something about (this is just a very basic list) –


The Choregraphe simulation environment -
[http://doc.aldebaran.com/1-14/software/choregraphe/index.html](http://doc.aldebaran.com/1-14/software/choregraphe/index.html)

There are some nice tutorials in YouTube that can get you started quickly, see for example
[https://www.youtube.com/watch?v=5pMx-K7AnEA](https://www.youtube.com/watch?v=5pMx-K7AnEA)

Also if you get stuck, you can get in touch with your team mentor.
3. **Installing the Software**

From 2017, the Choregraphe software can be freely downloaded from this link-

https://developer.softbankrobotics.com/us-en/downloads/pepper (for windows, use version 2.5.5)

All the labs will already have it installed, but you can also install it in your own laptops (if you want to work on the robot remotely from your home).

When you open choregraphe, you will see something like this...

4. **Learning to use basic/built in features with simulated robot**

By default you are connected to the virtual robot (i.e. the robot in the simulated environment). The simulated robot allows you to test basic behaviors- before working on the real robot.

To connect to the real robot you need to enter its IP Address (someone will be there to help you with that). In general if you press the button on the chest of the robot, it will tell its name and its IP address.

You might start playing with simple existing behaviors. For example, using the sit down box will make the robot sit down
Note that to design the Basic MarcoPoloBot you don’t need anything other than existing blocks-though you need to figure out which blocks you will need, how to connect them, what parameters to pass.

Incorporating additional features in your design, might need some minor programming skills.

You can also look at some tutorials like speech recognition- https://www.youtube.com/watch?v=5pMx-K7AnEA

5. Thinking/Designing MarcoPoloBot system on paper
Once you become familiar with the building blocks, how to trigger them you need to start thinking about the overall design of your MarcoPoloBot:

- What building blocks are needed
- How to connect them
- What parameters need to be passed
- How will the demo look like
- Any additional features you can think of
- Come up with a block diagram in paper

6. Implementing the Design, Testing MarcoPoloBot on the Real Robot
Make sure that you test individual subsystems separately before you test the full system- it always helps when you are designing a complex system, makes debugging easier. When you are ready to test on the real robot, contact the GLA or Dr. Mohan and they can give you access to the robot.
8. **Additional Functionalities’ (open to your creativity)**

In addition to the basic MarcoPoloBot, you may think of other features that will make your demo more impressive, like

- Make the robot more interactive
- Once the robot reaches a continent, it could say some information about that continent
- The robot can move from one continent to another remembering where it is presently and find the correct path to go to the next continent (for example, from Europe to Africa)
- Other ideas...
GAMES CHALLENGE

CE101
10TH OCTOBER, 2017

PROF. RICHARD A. BARTLE
• So, **who** are we and **why** are we here?

• I’m **Richard Bartle** Bsc Phd FRSA FBCS
  – Honorary Professor of Computer Game Design
  – Just call me **Richard**

• I teach **two modules for the computer games degree**
  – **Computer science** students can take the 2nd-year one (CE217)
  – They can also take the 3rd-year one (CE317)

  • But **only** if they took the 2nd-year one
YOU

• YOU ARE 1ST-YEAR UNDERGRADUATES IN THE SCHOOL OF COMPUTER SCIENCE AND ELECTRONIC ENGINEERING (CSEE)

• YOU HAVE NO IDEA WHAT'S GOING ON

• DON'T WORRY, NEITHER DO I
  – THIS IS THE FIRST TIME WE'VE TRIED THIS...

• MOST OF THE 25ISH PEOPLE ON THE GAMES DEGREE SHOULD BE IN THIS GROUP
  – I GUESS SOME ODDBALLS MAY HAVE LEFT

• THE REMAINING 35ISH ARE OTHER CSEE FOLKS
• **Using magic**, you have been allocated into **groups** by challenge topic
  – Which you had a **change** to exchange
• **I'm now going to let you swap groups within the games challenge**
  – Which may or may not be a **good idea**, depending on how sexist, racist etc. you are
• **All groups must have 6 members**
  – Maybe 5 if we don't have enough people
• **No, you can't be in a group of one...**
• SO, **FIND** your fellow group **MEMBERS** and **CLUMP** together

• WE’RE **HAVING** groups, by the way, so you get to **MEET** other people
  – even if you think they’re **WACKOES**, at least you’ve **MET** them

• OK, **ALL** in your **GROUPS**?

• RIGHT THEN: IF you want to **CHANGE** groups, do so **NOW**

• YOUR groups are now formally **FIXED**
The challenge is the **same** for every group over the next **three** afternoons:

- Each team has to create a **board game** on the theme of **Vikings**
- The board game has to be **entertaining**, yet **not involve 18-rated content**
- The game must be for **no fewer than 2 and no more than 6 players**
- The **majority** of the game’s board has to use **hexagons**
WHY A BOARD GAME?

• WHY A **BOARD** GAME?

• IT'S **FUN**!
  - OK, SO ALSO THERE ISN'T ENOUGH **TIME** TO DEVELOP A VIDEO GAME...

• BOARD GAMES ARE IMPORTANT **TOOLS** USED BY VIDEO GAME **DESIGNERS** FOR TESTING AND **PROTOTYPING** IDEAS
  - EASY TO MAKE, PLAY, **TEST**, REDESIGN, REPLAY, ...

• THEY INVOLVE MULTIPLE INTERACTING SYSTEMS
  - SO ARE INTERESTING TO **PROGRAMMERS**, TOO
WHY VIKINGS?

• Why a theme of VIKINGS?
• Vikings are COOL!
• Everyone has heard of VIKINGS!
  – We name our WEEKDAYS after their GODS!
• They’ve influenced much European culture
  – Widely enough to avoid concerns regarding cultural APPROPRIATION
• They were ACCOMPLISHED in many fields
• If I’d left the topic OPEN, you’d all have made post-apocalypse ZOMBIE games
**WHY ENTERTAINING?**

- **WHY SHOULD THE GAME BE ENTERTAINING?**
  - Well, it’s a **GAME**
    - Plus it’ll be more fun for you to **playtest** your game if it’s **entertaining**
  - **WHY NO 18 AGE RATING?**
    - Vikings have a licentious reputation, after all
  - **JUST BECAUSE YOUR FAMILY ISN’T WATCHING YOU BREACH SOCIAL NORMS, THAT DOESN’T MEAN YOUR FELLOW STUDENTS AREN’T**
  - **LET’S KEEP IT CLEAN, PLEASE**
WHY HEXES?

• WHY HAVE HEXES ON THE BOARD?

• I DON’T WANT YOU RESKINNING AN EXISTING GAME

• FEW OF YOU WILL HAVE PLAYED GAMES WITH HEX BOARDS BEFORE
  – THE GAME-LITERATE AMONG YOU MAY HAVE, BUT YOU’RE LESS LIKELY TO RIP OFF GAME IDEAS

• NOTE THAT THE BOARD ITSELF DOESN’T HAVE TO BE HEXAGONAL
  – IT JUST NEEDS THE MAIN PLAY SPACE TO USE THEM
Why 2-6 players?

• I want other groups to be able to play your game
  – so the upper limit is 6
• Single-player games tend to be puzzles or unreplayable
  – so the lower limit is 2
• You can have something in between
  – for example, 3-4 players or 2-player only
• You can have different rules for different numbers of players
THE DELIVERABLE

• SOME TIME TOWARDS THE END OF THURSDAY I’LL WANT TO SEE A STURDY GAME PLUS RULEBOOK
  – RULES CAN BE HANDWRITTEN IF THEY’RE LEGIBLE

• OBVIOUSLY THE SET DOESN’T HAVE TO BE OF PROFESSIONAL QUALITY, BUT IT DOES HAVE TO BE NON-OFFPUTTING

• OTHER GROUPS WILL GET TO PLAY YOUR GAME
  – BUT HEY, YOU’LL GET TO PLAY THEIRS!
• This is a game I made when I was 12
  – (Iterations 3-5, so I was 15+ by then)
• One of the **reasons** I decided on a board game is to familiarise you with **stuff**

• **All** professional game designers have a **stash of stuff** they use for testing and prototyping game designs

• I spent **£453** of the department's money buying stuff for you to use
  – So like **9 lectures'** worth of your **fees**

• Don't be **greedy** as it is communal
  – I will take stuff **off** you if you **hoard** it
Here's what I ordered (Gawd knows if it all arrived):

- A4 paper, assorted colours
- A3 paper, white
- A4 sheets of hex paper, white, various gauges
- A4 card, assorted colours
- Blank playing cards
- Address labels (for sticking on playing cards)
- A4 foamboard (for backing final board)
- ICM cubes, assorted colours
STUFF LIST PART 2

- 8mm adhesive dots, assorted colours
- 13mm adhesive dots, assorted colours
- A4 funky foam (for making pieces)
- Scissors
- Heavy-duty card-cutter
- 25mm sticky tape dispenser (plus tape)
- 5mm corner rounders
- 25mm circle punch
- 6-sided dice, assorted colours
- 10-sided dice, assorted colours
STUFF LIST PART 3

- Long-arm stapler
- Tiddlywinks, assorted colours
- Meeples, assorted colours
- Plastic pawns, assorted colours
- Play gemstones
- Plastic treasure coins
- C4 cardboard boxes (to box up your game)
- Elastic bands, assorted colours and sizes
- Viking stickers
- Viking sticker books
**Absent**

- **Absent from the list are:**
  - **Pens, pencils and drawing implements that you should have anyway**
  - **Sticky-backed Plastic, because you’ll despair when it goes horribly wrong**
  - **A craft knife and safety ruler (let’s keep those fingertips attached)**

- **Don’t feel you have to work here**
  - Although here is where the stuff is
  - Feel **free** to use your **own** stuff if you want, of course
LEARNING OUTCOMES

• You’ll learn what it takes to make a board game

• You’ll learn useful prototyping skills

• You’ll understand the difference between a test game and its final version

• You’ll learn whom to hang out with and whom to avoid
  – and that group projects always suck

• You’ll learn that your opinions and passions about games are shared
MARK SCHEME

• WE’LL PROBABLY ROLL DICE OR SOMETHING
• BASICALLY, YOU ARE MARKED AS A GROUP
  – EXCEPT THOSE WHO DON’T PULL THEIR WEIGHT, WHO’LL BE LOOKING AT A POTENTIAL ZERO
• WE’RE BASICALLY TRYING TO GIVE YOU 10% OF YOUR FIRST-YEAR MARKS THIS WEEK
• IT’S ONLY THREE DAYS
• FOR 10%, YOU CAN STAND THE COMPANY OF COMPLETE STRANGERS FOR THREE DAYS
Advice

- Don’t start off making a polished version of the game
  - Go through several iterations, of which only the last will be polished
- Understand your audience
  - It doesn’t have to be students, by the way
- Try to figure out what your game is saying to its players
  - Once you know that, everything else will flow from it
• I’ll be wandering around to see what you’re doing
  – And taking names so I know who’s slacking
• One or more PhD students will also be here (Joseph and Piers)
  – And not only to stop you nicking the stuff
    and eBaying it
  – Note that they are actually gamers!
• Any questions?
• OK, off you go!
And finally...

- **TOMORROW’S SESSION WON’T BE HERE, IT’LL BE IN THE TEACHING CENTRE:**
  - Rooms **TC2.6 AND TC2.7, MUSHED TOGETHER**

- **THURSDAY’S SESSION WILL BE IN NTC 3.05**
  - **IT’S ALMOST AS IF NO-ONE PLANNED ANY OF THIS IN ADVANCE**
    - **TO BE FAIR, THEY DID, BUT IT WAS GOING TO BE IN A LAB WITH COMPUTERS IN THE WAY**
Receiving ADS-B Transmissions

Adrian F. Clark
(alien@essex.ac.uk)

2017/08/09 at 13:05:25 +01'00'

School of Computer Science and Electronic Engineering, University of Essex
Colchester CO4 3SQ
1 Introduction

Modern commercial aircraft are equipped with radio beacons that regularly broadcast information about the aircraft itself and its flight. As the eastern approach to Stansted runs near Colchester, it is fairly easy to pick up these transmissions, given suitable radio equipment. A few years ago, radios that could do this were pretty expensive; but recently, people discovered that some of the USB ‘dongles’ used to receive digital TV also receive in the waveband used by these radio beacons.

The aircraft beacons transmit using a protocol called ADS-B. Transmissions contain a digital packet — much more like wifi than, say, AM or FM radio broadcasts — meaning that a computer is needed to decode them. Your job is to build a system based around a Raspberry Pi, a single-board computer, that receives and displays these transmissions. In doing this, you will gain the following skills:

- finding information on the web
- constructing a computer from components
- connecting a computer to the campus network
- connecting hardware to computers
- installing the operating system and applications on a Raspberry Pi
- what Linux is and a little of how to use it
- the relationship between frequency and wavelength
- designing an antenna for a particular frequency
- understanding that different antennae have different properties
- packet-based transmissions

Some of these are specific to this project while others are generic, able to be applied to any task you undertake in the future.

One of the things that you will find in a University is that the instructions you receive will be less detailed than you have been used to in the past — and will become less detailed as you progress through the degree. This is because one characteristic of a good graduate is the ability to work out the detailed steps from a fairly general specification. This is a difficult skill to teach and the idea is that you learn it through experience, so don’t be worried if you get stuck — and when you do, be sure to ask for help, first from the other members of your group and then from the postgraduates and academics helping with this project. You are here to learn and asking questions is often the fastest way to do that.

You will work in groups of about six people. Each group will be provided with the following equipment:

- a Raspberry Pi single-board computer in a box
- a micro-SD card and a SD-sized carrier for it
- a power supply, keyboard and mouse
- a USB radio receiver dongle and aerial
- an Ethernet cable and a HDMI cable
- some connectors

It is up to you how to arrange your group: you might decide that each member of the group should work on a specific task (but remember that someone needs to be responsible for coordinating tasks!) or you might all wish to work together. Try to make a conscious decision about the way you work, and remember why you came to that decision for later in this module.
Questions
Before proceeding, you might like have a go at finding the answers to the following questions:

1. What does ADS-B stand for?
2. What frequency is used for ADS-B transmissions?
3. What do ADS-B transmissions contain?

2 Downloading the operating system

As you probably know, the Raspberry Pi doesn’t run Windows but rather a version of Linux known as Raspbian.[1] As with most Linux distributions, Raspbian is free software. It provides similar functionality to Windows, though not in exactly the same way. The first step in your project is to find an up-to-date version of it—you want one that comes with desktop, which means with a graphical user interface (GUI). Find a handy Windows machine in one of CSEE’s computer labs and have one of your group download it.

The next step is to transfer the downloaded version of Raspbian, which is normally a zip-file, onto your group’s SD card. You cannot just drag and drop the file onto the SD card, nor can you unpack it and drag the resulting folders: this is because the actual format of the SD card under Linux differs from that used by Windows. Why should that be? One reason is that Windows filesystems lack useful features such as journalling, so what is convenient for Windows is not necessarily the best for other systems. You’ll find that most machines in CSEE have a piece of software installed on them called Etcher, with it, all you need do is identify the zip-file containing Raspbian, the SD card you want it written on, and then just get it to ‘burn’ (copy across) the file in the right way.

Questions

1. What is the format of an SD card as used by Windows?
2. What is a journalling filesystem?
3. Can you find a procedure for writing the SD card that doesn’t use Etcher?

3 Connecting up the hardware

Before building up your system, you need to know where to build it! In principle, you can do this pretty much anywhere there is a mains supply, a monitor and access to the campus network, but a good place would be CSEE’s Lab 8 (the Hardware Lab, located one floor down from the podium level on square 1). That lab is managed by Ian Dukes, so find yourself somewhere in it for your group to work that doesn’t impede other groups and Ian says is OK.

It should be pretty obvious how the Raspberry Pi goes together. If it isn’t, there is plenty of information to be found on the Web; and you can ask a demonstrator to look over your configuration before powering it up. At this point, it’s best not to plug in your USB radio dongle.

Although a Raspberry Pi is a reasonably robust piece of kit, it does contain delicate electronics and so some simple precautions should be observed: don’t use it on anything that conducts; don’t touch it with your body or anything metallic while it is switched on; and do keep it dry.

---

1Linux is an implementation of the Unix operating system, as is MacOS (which runs on Apple computers but not Apple phones) — though the Linux graphical user interface does not look the same as the Mac one unless you fiddle with it. Thankfully, the Raspbian GUI is not the same as the default one encountered on Ubuntu, the most popular Linux distribution, which is truly awful.
Questions

1. What is the type of connector between your Raspberry Pi and monitor?
2. What is the maximum amount of current that your Raspberry Pi will draw?
3. Why is it a bad idea to let electronics get wet?

4 Booting and upgrading Linux

With your SD card carrying Raspbian inserted and everything connected up, you simply apply power to the Raspberry Pi. If you’ve done everything right, you’ll see a series of messages on the screen, then a graphical panel proclaiming “Welcome to the Raspberry Pi Desktop” appear. After a minute or two, this will be replaced by the normal Raspbian graphical interface (Figure 1). Linux requires that each individual user has a unique name (the jargon is ‘user,’ ‘username’ or ‘login’); by default under Raspbian, this is pi and Raspbian is unusual in that it automatically logs this user in. This is a useful feature for inexperienced users (‘newbies’).

The first thing to do is upgrade your machine so that it has the latest version of the operating system on it. You do that by bringing up a terminal window and typing two commands in it:

```
sudo apt-get update
sudo apt-get upgrade
```

A few words of explanation are called for here. The `apt-get` command controls Linux’s package manager, the piece of software used for installing operating system packages (components). This has a record (the jargon word is ‘cache’) stored locally on the SD card of what versions of which packages are available, so the `update` command searches software repositories around the Internet for the most up-to-date versions and updates this cache; you’ll see output from the command reporting where it is looking. The `upgrade` command then downloads and installs the latest versions of all
the packages that came with the version of Raspbian you wrote onto the SD card. This will take several minutes and you'll probably be asked to approve the upgrade before it starts.

Why are both these commands prefixed by `sudo`? Installing packages is normally done by the system manager (the Unix jargon is ‘super-user’ or `root`, equivalent to the Administrator on Windows), not rank-and-file users. The `sudo` command makes the person who is logged in ‘become’ `root` for the duration of a single command. You can’t `sudo` on any of the computers in our software labs, of course.

The next question that most people ask themselves is why is it necessary to type commands? Isn’t that so 20th century? Well, no: everyone knows that experienced computer users type commands rather than use a GUI (see https://www.youtube.com/watch?v=xaVgRj2e5_s from about 2:45 minutes in). More seriously, you’ll find as you gain more experience that it’s easier to make the machine do exactly what you want using the command line. The author of this document works this way all of the time, for example.

Before moving on, a few words about the campus network. You may already have found that you can use your normal campus credentials (username and password) to get your computer or phone onto Eduroam, the campus wifi network. You’ll find if you visit other universities in the UK that you can use Eduroam there too, a really useful facility. However, there are a lot of things you cannot do from a machine connected via Eduroam, and for those you need to be on a machine on the campus’s wired (Ethernet) network. You can connect any computer to the campus Ethernet and it will work for that day; but it will be blocked overnight and will not work again subsequently unless you register the machine. All the Raspberry Pis used for this project are already registered with the campus network, of course.

Questions

1. What does the `apt` in `apt-get` stand for in?
2. What is the typical transfer speed of a wifi network?
3. What is the typical transfer speed of the campus Ethernet?

5 Installing the TV dongle drivers

If you have successfully followed the procedure outlined above, you now have a working, up-to-date Linux system installed on your Raspberry Pi. The next step is to install the ‘drivers’ (the low-level software) for your TV dongle. The following commands do that. You’ll see that the first of these is another `apt-get` command, so it’s installing software that wasn’t part of the initial version of Raspbian from the package repositories on the Internet. However, the software for handling the TV dongle isn’t a standard part of Raspbian, so the following command, the `git` one, downloads its source code while the remainder build and install it.

This is a bit of a fingerful to type and you need to do it exactly as shown below. Some of the commands will take some minutes to execute, so be patient.

```
sudo apt-get install libusb-1.0.0-dev libmysqlclient-dev cmake
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
make.
make
sudo make install
sudo bash
cd /etc/modprobe.d
cat <<EOF >rtl-sdr-blacklist.conf
blacklist dvb_usb_rtl28xxu
EOF
```
This is the most difficult part of the whole exercise. After you have done this, type the command `reboot` to re-boot your Raspberry Pi. While it is doing that, plug the TV dongle into a USB port.

Questions

1. If you watched the compilation commands as the software was built, can you figure out what programming language the software was written in?
2. What is `git`?
3. SDR in the above commands stands for ‘software-defined radio’? What does that mean?

6 Installing the ADS-B software

The final part of the software process is to install the software, `dump1090`, that receives and makes sense of the ADS-B transmissions. The first thing to do is find the source code, which is stored on a website called GitHub. Crank up a web browser and search for “Malcolm Robb dump1090”. You should eventually end up on a page that looks a bit like the one in Figure 2. Click the green button to download the zip-file, which will end up in your `Downloads` folder (the Linux jargon is ‘directory’).
You then unpack and build the software (by typing commands into a terminal window, of course) as follows:

```
unzip ~/Downloads/dump1090-master.zip
cd dump1090-master
make
```

and twiddle your thumbs for a few minutes while it compiles.

7 Running dump1090

Having built the software, all that remains is to run it:

```
sudo bash
export LD_LIBRARY_PATH=/usr/local/lib
./dump1090
```

At the time of writing, there is a problem with the version of Raspbian you have installed and the preceding commands work around that problem. If everything has gone well, your terminal will start displaying ADS-B messages similar to those shown in Figure 3. However, this isn’t the best way to use dump1090. Interrupt it by typing ^C (hold down the Control or Ctrl character on the keyboard and then hit C) and run it again with some ‘qualifiers’ that change the way it operates.

```
./dump1090 --interactive --net --modeac --phase-enhance --aggressive
```

This tells dump1090 to work as a web-server on port 8080.

Then start your web browser and point it at http://localhost:8080. A map of Europe should appear; zoom in on Colchester and you’ll see that an icon is drawn on it for each aircraft whose transmissions are being received. A typical appearance of the screen is shown in Figure 4. If you click on an aircraft icon, the recent route of it will be drawn on the map and some of its details will be presented in the upper right part of the page.

Figure 3: Typical output from dump1090

---

2For the initiated reader, it is that the library search path is configured to include /usr/local/lib but for some reason doesn’t.
Questions

1. What do the various qualifiers (the parts that start with `--`) make `dump1090` do?
2. Why does it have the 1090 in its name?
3. What port does web traffic normally use?

8 Improving reception

The aerial (sometimes called an ‘antenna’) that comes with the TV dongle works but isn’t optimised for receiving ADS-B transmissions. In the last part of the project, you have an opportunity to try out antennae that are a good match to the transmissions and to ascertain how well they work.

8.1 Tune the supplied antenna

You will hopefully have realised by now that the frequency used for the ADS-B transmissions is $f = 1090$ MHz. Knowing that the speed of light is $c = 3 \times 10^8$ m/s and that the wavelength of radio transmissions $\lambda$ is related to these by $c = f\lambda$, you should be able to work out the corresponding wavelength. Theory suggests that the best length of your aerial should be $\lambda/4$, so does changing it have any effect on your reception?

8.2 Making a dipole

A dipole is the simplest antenna. It consists of two straight pieces (‘arms’) which are $\lambda/4$ in length. One arm is connected to the centre conductor of the coaxial ‘feeder’ cable that runs from the dongle while the other is connected to the outer metal shielding. Your group can get a feeder cable, copper wire and corks from Ian Dukes and make up a dipole, then try it with your receiver system. Does
mounting the dipole horizontally or vertically make any difference? If you make a little coil of three or four turns of the feeder right next to the dipole, does that make any difference to reception?

8.3 Making a yagi

The next step up in complexity from a dipole is a yagi antenna. You will have seen this type of antenna before: it is normally used for terrestrial TV reception in the UK. Taking your dipole as the starting point, it adds two parallel lengths of wires, spaced $\lambda/4$ to each side of it; the first of these is $1.05\lambda/2$ and the other $0.95\lambda/2$. The longer wire is a kind of reflector and the shorter helps direct the radio onto the dipole.

If you try your yagi out by pointing it roughly east, you should find that it is able to receive transmissions from further away than the standard aerial or your dipole; however, you will lose the ones to the west more quickly. This is because the yagi is a directional antenna.

Questions
1. Can you find out what the little coil of wire alluded to above is called?
2. Can you find any other antenna types for receiving ADS-B transmissions?
3. If there any other type of feeder cable for aerials apart from coaxial?

9 Concluding remarks

This project has hopefully got you up and running in a number of things. Although you have worked on a technical problem, the most important skills it has tried to give you are working with others and not being afraid to ask questions! It will also have introduced you to a few other people in your year.

At a technical level, the project has got you to use a type of computer that will most likely be unfamiliar, both in terms of hardware and software. If you think that Linux is an oddity, best suited to ‘toy’ computers such as the Raspberry Pi while proper computing is done on Windows, you couldn’t be more wrong: most of the world’s fastest computers use Linux (though not Raspberry Pis!), and much of the Internet is based around computers that run it.

Working with different antennae has given you a little introduction to electromagnetics. Only Electronic Engineering students study that topic in detail (and most definitely not in their first year!) but it should be clear from your simple experiments that the length and orientation of an antenna affects the quality of reception.

If doing this project has made you curious about radio, you might be interested to learn that Essex has an active amateur radio society, EARS, and its members are able to send messages to other radio amateurs in most countries around the globe. To be able to broadcast on the amateur bands, you need to hold a licence from Ofcom, the UK telecommunications regulatory body; EARS regularly runs courses that let you obtain a licence — and you can do this even if you are a non-UK student. To find out more, contact the author of this document.
Chatterbots

Introduction to chatterbots

The following passage is taken directly from Wikipedia:

In 1950, Alan Turing published his famous article "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence. This criterion depends on the ability of a computer program to impersonate a human in a real-time written conversation with a human judge, sufficiently well that the judge is unable to distinguish reliably—on the basis of the conversational content alone—between the program and a real human. The notoriety of Turing's proposed test stimulated great interest in Joseph Weizenbaum's program ELIZA, published in 1966, which seemed to be able to fool users into believing that they were conversing with a real human.

Eliza consisted of a file containing a series of key words and/or sentences, with an associated response sentence, for example if the user typed: “hello what a lovely day” Eliza would find a match for the word hello and respond with “hi how are you doing” thereby creating the illusion of intelligence although all that has taken place is the matching of a keyword.

More recent Chatterbots include ALICE which uses XML and typically contains in excess of 50,000 sentences in its database. What makes ALICE particularly clever is the ability to decompose sentences to produce multiple responses and extract and remember the context of the conversation.

There is a competition called the Loebner Prize which runs annually and broadly responds to the Turing challenge. The current 2017 award is held by a machine called Mitsuki by Steve Worswick.

Writing your own Chatterbot

Taking the basic concept of Eliza we can write a very simple Chatterbot of our own. My suggestion is to do it in Python but if you wish to try another language our staff will help you if it is feasible. Here are some suggestions:

(i) Create a text file that contains two columns, one column should contain the keyword, and the other the response, e.g.

Hello <break> welcome to the programming course
In the example here the keyword is separated by the control word `<break>`. This makes it easier to write code to parse the data

(ii) Another way is to create a two dimensional array of strings or a list of tuples

```
Database = [
    ['hello', 'welcome to the programming course'],
    ['date', 'it is the 13th July 2012'],
    ['name', 'my name is Fred'],
    ['weather', 'it is always sunny At Essex'],
]
```

(iii) Once a simple 'database' has been created it needs to be searched using a keyword. The keyword needs to be input from the command line by the user, this can be done in Python using:

```
variable = raw_input("please enter a keyword")
```

(iv) Once the user input has been captured it can be searched for in the database. Assuming the database has been defined as a 2D array as in (ii) using the following code will search for an occurrence of the single key word.

```
for i in range(4):
    if Database[i][0] == variable:
        print Database[i][1]
```

The full code listing is shown below. You should note that the code is case sensitive.

**Run the Python GUI called Idle which can be found under ‘all programs, Python31’.** Your demonstrators will walk you through this on the board

*Listing for diy_chatterbot.py*

```python
Database = [
    ['hello', 'welcome to the programming course'],
    ['date', 'it is the 13th July 2012'],
    ['name', 'my name is Fred'],
    ['weather', 'it is always sunny At Essex'],
]

variable = raw_input("please enter a keyword")
```

**Written by Dr John Charles Woods September 2017**
for i in range(4):
    if Database[i][0] == variable:
        print Database[i][1]

variable = raw_input("press enter to terminate")

Things to think about:

(i) Put the input into an infinite loop, conversations are more than one word
(ii) increase the size of the database
(iii) allow the input of sentences rather than single words
(iv) solve the problem of case sensitivity
(v) solve the problem of not all words matching in a sentence
(vi) solve the problem of spelling mistakes and typos
(vii) Employ voice recognition and synthesis
(viii) identify important and non-important words

Here is a modified version which allows multiple questions and the keywords are more meaningful:

Database = [
    ['hello', 'welcome to the programming course'],
    ['what is the date', 'it is the 13th July 2012'],
    ['what is your name', 'my name is Fred'],
    ['what is the weather', 'it is always sunny At Essex'],
    ['how old are you', 'mind your own business, how old are you?']
]

while 1:
    variable = raw_input("> ")

    for i in range(5):
        if Database[i][0] == variable:
            print Database[i][1]

    The next stage performed in chatterbots, which you are not expected to do is sentence decomposition, which breaks a long sentence down in to manageable chunks for the piece of software

    The Turing Challenge and the Loebner Prize are now within your grasp ..................

Written by Dr John Charles Woods September 2017