University Carlos III of Madrid

Industrial Electronics and Automation Engineering

Bachelor Thesis

Design, construction and programming of a low cost, Open Source robot for assistive activities

Author
Alvaro Ferrán Cifuentes

Supervisor
Juan González Víctores

Contents

List of Figures

Listings

Part I
Introduction

1 Introduction

2 Socio-economic factors

ageing population, prevision spain, social robots, etc
Android: great market share, cheap
Arduino, raspberry: very cheap 20+35 vs 300 normal pc
3d printer vs injection- / kg vs / kg

3 Regulatory compliance

complies with normativa etc y normativa etc2

4 Scope of the project

5 Project phases

[include Gantt diagram]

Part II
State of the art

Part III
Proposed Solution

6 Project components

6.1 3D Printer

3D printers are Computer Numerical Control (CNC) machines that are capable of transforming virtual 3D models created with a Computer Aided Design (CAD) software into real-world objects.

Created in 1984 by Chuck Hull of 3D Systems Corp this technology was little-known to the general public and was mainly used in industries for short runs of difficult pieces.
In 2005 Dr. Adrian Bowyer, from the University of Bath, UK, started the RepRap project. Its goal was "to produce a pure self-replicating device not for its own sake, but rather to put in the hands of individuals anywhere on the planet, for a minimal outlay of capital, a desktop manufacturing system that would enable the individual to manufacture many of the artifacts used in everyday life"

Today a vast range of 3D printers co-exist, varying in size, price and materials used.

→ [RepRap, Zcorp, chocolate, liquid sint, micro, house building]

→ [table with different methods?]

In this theses a RepRap Prusa Air 2 model is used. It is of a "fused filament fabrication additive manufacturing" type. This type of printers extrude mainly ABS or PLA plastics, and deposit new liquified material over ther previous layer, now solid, effectively building parts from the bottom up layer by layer.
→ **Terminal Imprusión Autorreplicante (T.I.A)** (add info + pic) Built within the scope of the Clone Wars project

6.2 Software

This type of 3D printers work by turning 3D models into plastic parts. These models are first modelled in a CAD program and then processed with a *slicing* software to divide the model into layers of G-code, which is the standard language interpreted by CNCs. This is then introduced in a third piece of software which feeds it to the printer.

6.2.1 3D Modelling

In this project Sketchup has been used to create the printed parts. Owned by the company Trimble Navigation it is a WYSIWYG (What You See Is What You Get) modelling editor with a large online warehouse of parts available for download.

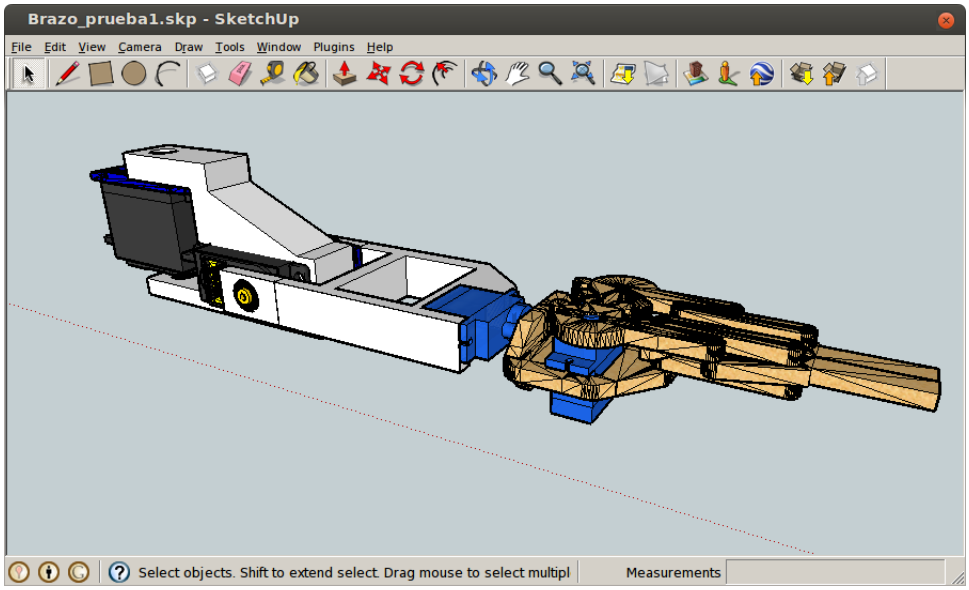

Figure 1: SketchUp software

In order to make it compatible with the slicing sofware, Sketchup's propietary format, *SKP*, has to be converted to the standard *STL*. In order to accomplish this the *Su2stl.rb* plugin is installed. A new *Plugins* menu appears in Sketchup which contains the Import/Export options, where the desired output format and model units are specified.

6.2.2 G-Code Generator

Once the model is converted to *STL* it then has to be sliced. Since 3D printers work by building layer upon layer of plastic, the model has to be transformed into the same format. The G-code generator converts the CAD model into layers of CNC instructions. There are three main slicing programs, each with their own benefits:

- Skeinforge:
  The first slicing program used in homemade 3D printers. It is by far the most complete of the three. It allows the user to control each and every imaginable setting of the printer, from the axis' speeds to the retraction distance of the plastic into the extruder while moving. However, because of this it has a very steep learning curve which makes it unsuitable for the average consumer.

- Slic3r:
  Slic3r was created as an user-friendly software, which only gives the final user a choice in the basic settings, such as printing speeds, filament widths or part infills. As a result it is an easier program to slice parts with a sufficient level of customization. It has nonetheless problems converting models with imperfections or broken shapes.

- Cura
  Finally, Cura is also designed with user-friendliness in mind. This slicer is more robust than Slic3r, in that it will accept models with imperfections, and will try to correct them. It also features a box simulating the print area in which the model can be moved around, turned or scaled before printing. This last feature is specially useful if minor changes need to be made, without returning to the CAD software.

6.2.3 CNC Controller

6.3 Li-Ion Battery

The whole system is powered by a lithium ion 12V 6800mAh battery.

Figure 2: Li-ion 12V 6800mAh battery with charger

Many different types of batteries are available in the market, each with their own benefits and disadvantages. The most common types are alkaline, lithium ion (Li-ion), lithium polymer (LiPo), lead acid, nickel–metal hydride (NiMH) and nickel–cadmium (NiCd).

	NiCd	NiMH	Lead Acid	Li-ion	Li-ion polymer	Reusable Alkaline
Gravimetric Energy Density(Wh/kg)	45-80	60-120	30-50	110-160	100-130	80 (initial)
Internal Resistance (includes peripheral circuits) in mΩ	100 to 200[1] 6V pack	200 to 300[1] 6V pack	<100[1] 12V pack	150 to 250[1] 7.2V pack	200 to 300[1] 7.2V pack	200 to 2000[1] 6V pack
Cycle Life (to 80% of initial capacity)	1500[2]	300 to 500[2,3]	200 to 300[2]	500 to 1000[3]	300 to 500	50[3] (to 50%)
Fast Charge Time	1h typical	2-4h	8-16h	2-4h	2-4h	2-3h
Overcharge Tolerance	moderate	low	high	very low	low	moderate
Self-discharge / Month (room temperature)	20%[4]	30%[4]	5%	10%[5]	~10%[5]	0.3%
Cell Voltage(nominal)	1.25V[6]	1.25V[6]	2V	3.6V	3.6V	1.5V
Load Current						
- peak	20C	5C	5C[7]	>2C	>2C	0.5C
- best result	1C	0.5C or lower	0.2C	1C or lower	1C or lower	0.2C or lower
Operating Temperature(discharge only)	-40 to 60°C	-20 to 60°C	-20 to 60°C	-20 to 60°C	0 to 60°C	0 to 65°C
Maintenance Requirement	30 to 60 days	60 to 90 days	3 to 6 months[9]	not req.	not req.	not req.
Typical Battery Cost (US$, reference only)	$50 (7.2V)	$60 (7.2V)	$25 (6V)	$100 (7.2V)	$100 (7.2V)	$5 (9V)
Cost per Cycle(US$)[11]	$0.04	$0.12	$0.10	$0.14	$0.29	$0.10-0.50
Commercial use since	1950	1990	1970	1991	1999	1992

Figure 3: Table comparing different battery technologies

6.4 Voltage level converters

6.4.1 DC-DC Step-Down Converter

Voltage level converters are used to adapt a source's voltage to that required by the load. In this thesis a DC-DC converter is used to decrease the 12V given by the battery to the 5V required by the logic components as well as the servomotors.

The simplest method would be to use a linear regulator such as a 7805, which is a cheap, single-component solution. However, this is greatly inefficient solution, since a great part of the power is dissipated as heat. For instance, if a 7805 were to be used in this project, about $\frac{Power_{in}-Power_{out}}{Power_{in}} = \frac{12 \cdot I - 5 \cdot I}{12 \cdot I} = \frac{12-5}{12} = 58.33\%$ of the power is wasted.

A much more efficient solution is to use a switching regulator such as a Buck converter, which has an efficiency level of around 95% [1]. Buck converters work by switching rapidly between "On" and "Off" states, which sets the output voltage in function of the duty cycle $d = \frac{time_{on}}{time_{on}+time_{off}}$. The converter used includes a potentiometer to set the output level by selecting the duty cycle.

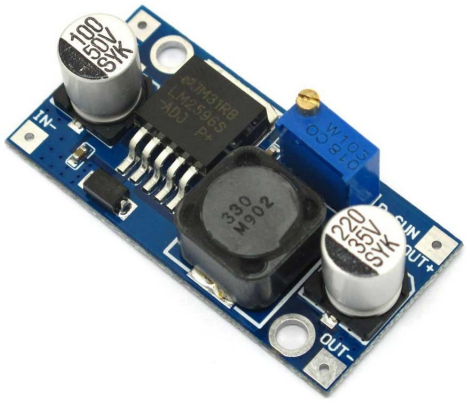

Figure 4: LM2596S step-down converter

The converter's electrical specifications are:

- Adjustable input voltage: 3.2 - 40V
- Adjustable output voltage: 1.25 - 35V ($V_{in} > V_{out}$ + 1.5V)
- Max. output current: 3A

6.4.2 Bidirectional logic level converter

In order to enable serial communication between the Arduino and the Raspberry Pi another voltage level converter must be introduced, since the former operates at a 5V level while the latter does so at a 3.3V level.

In this case a switching regulator like the previous one will not work because the communications are much faster than the regulator's switching speed. Therefore a bidirectinal, low power converter can be built out of transistors.

Figure 5 shows a simple converter model. Analyzing the circuit from the low side:

- If a logic one is emitted, the transistor source pin is grounded and it switches on, pulling down the high side to zero.
- If a logic zero is sent, the transistor is tied high and so is off, leaving the high pin connected to the pull-up resistor and thus seeing a one.

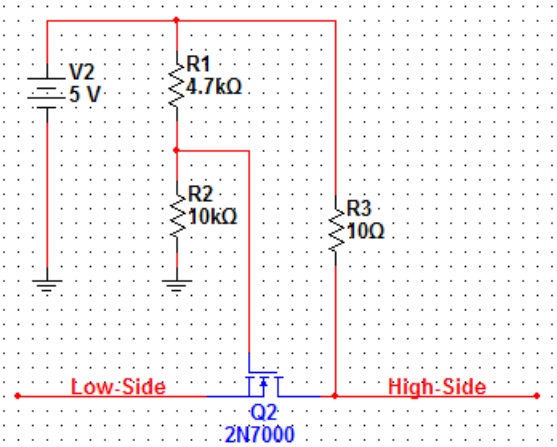

Figure 5: Bidirectional voltage converter

This setup works for one line, two identical circuits are needed in order to provide for serial communication. For this project a commercial board is used to reduce the total size of the converter by using SMD components.

Figure 6: JY-MCU 5V-3V converter

This board is used with UART communication, but is equally adequate for I^2C, SPI or one-wire communication.

6.5 Motors

6.5.1 DC motor

Figure 7: GA25Y370 motor

6.5.2 Servomotors

Figure 8: GOTECK GS-551MG servo

Figure 9: TowerPro SG90 servo

6.6 Arduino

Arduino is a family of low-cost electronic boards designed to be easily programmable. From the official Arduino website, "Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. "

Arduino is programmed using its own language, which is merely a set of C/C++ functions compiled with *avr-g++*. They can nonetheless be programmed in pure C or C++ in an external IDE and have code uploaded to it as any other AVR board.

In this proyect an Arduino Nano v3 with an ATmega 328 microcontroller has been chosen mainly due to its processing power and reduced size.

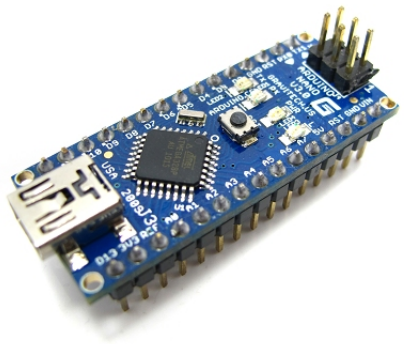

Figure 10: Arduino Nano v3

The official Arduino Nano V3 specifications are:

- **Microcontroller:** Atmel ATmega168 or ATmega 328
- **Operating Voltage (logic level):** 5V
- **Input Voltage (recommended):** 7-12V
- **Input Voltage (limits):** 6-20V
- **Digital I/O Pins:** 14 (of which 6 provide PWM output)
- **Analog Input Pins:** 8
- **DC Current per I/O Pin:** 40mA
- **Flash Memory:** 16 KB (ATmega168) or 32 KB (ATmega328), of which 2 KB used by bootloader
- **SRAM:** 1 KB (ATmega168) or 2 KB (ATmega328)
- **EEPROM:** 512 bytes (ATmega168) or 1 KB (ATmega328)
- **Clock Speed:** 16MHz
- **Dimensions:** 0.73" x 1.70"
- **Communications:** UART, SPI and I^2C buses

6.7 Raspberry Pi

From the official website of the homonymous foundation, the Raspberry Pi is a "credit-card sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects."

Figure 11: Raspberry Pi model B

Available in two models, A and B, the Raspberry has a Broadcom BCM2835 System On a Chip, which includes an ARM1176JZF-S 700MHz processor and a VideoCore IV GPU. It includes as well a 256Mb RAM, upgraded to 512Mb in model B.

The Pi features:

- HDMI, composite and raw DSI video outputs
- 3.5mm audio jack
- SD card socket
- Low-level peripheral connections including:
  - 8 General Purpose Input Output (GPIO) pins
  - Universal Asynchronous Receiver Transmitter (UART) bus
  - Inter-Integrated Circuit (I^2C) bus
  - 2 Serial Peripheral Interface (SPI) buses
  - Power pins: 3.3V, 5V and GND
- Ethernet socket
- USB hub (1 socket in model A, 2 in model B)

The main storage unit is the SD card, and that is where the OS is flashed, normally a Linux distribution. The most popular is Raspbian, an adapted version of Debian Wheezy, although other Linux distros or even other OS like Android or XBMC can be used.

(a) WiFi USB dongle (b) PlayStation 2 EyeToy (c) LCD screen 16x2

Figure 12: Raspberry Pi peripherals

In this proyect a Raspberry Pi model B running Raspbian manages the software side of the robot. It has an EDUP 802.11n WiFi USB dongle , a PlayStation 2 EyeToy USB camera and a 16x2 character LCD screen connected in order to create a WIFI Access Point, stream video to the user and signal its status respectively.

6.8 Android Phone

Android is... blablabla (history)
for this proyect the model bla bla haas been used

Figure 13: Android smartphone Haipai Noble H868

7 Hardware assembly

7.1 Mechanical structure

7.1.1 Upper body

denavit hartenberg for arms

7.1.2 Lower body

election of wheels vs legs

7.1.3 Assembly

cad models of assembly
list of all the parts with images images with assembly

7.2 Electrical connections

Figure 14 shows how the different electric and electronic components are interconnected. As it can be seen, different voltage levels co-exist within the robot, so regulators are placed to ensure the components function correctly.

The DC motors need the highest voltage to work, and so are connected to the battery, which provides them with the 12V they need. However they have to be controlled by the Arduino, hence the need for a driver that will turn on and off the 12V rails from 5V signals.

The rest of the components operate at 5V, which is why the step-down converter is used to convert the battery's 12V output into the desired level. The Raspberry Pi, Arduino and servomotors are connected to this rail.

Finally, the Raspberry communicates with the Arduino through the former's UART pins, which operate at a 3.3V level and can be damaged by the latter's 5V level pins. To avoid this a logic level shifter is introduced, which ensures data transmission without compromising the hardware's integrity.

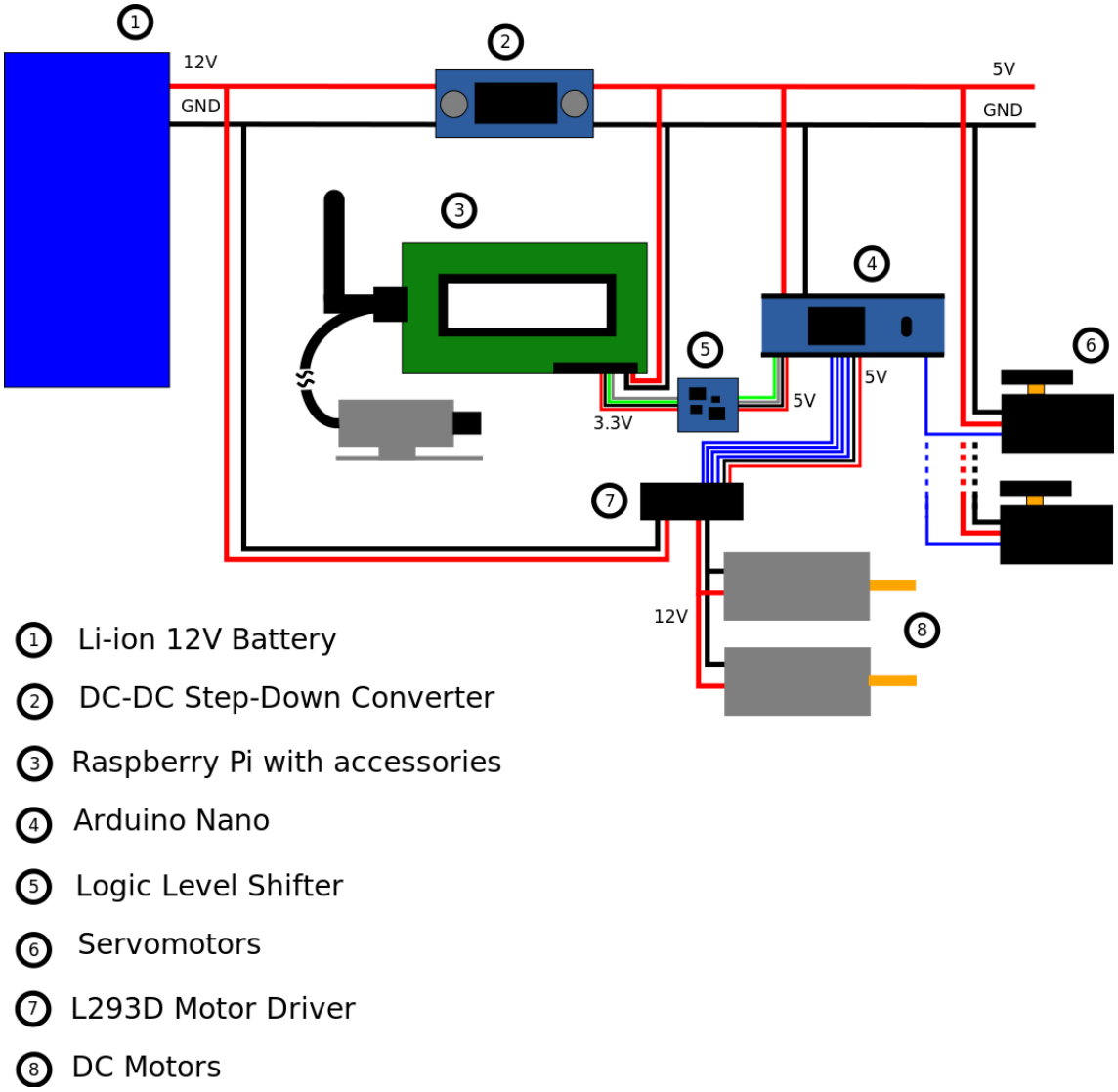

Figure 14: Electrical connections diagram

7.3 Logic connections

Figure 15 shows how the different components communicate between themselves. As it can be seen, the user controls the robot from the Android application. This implements a bidirectional communication over wifi with the Raspberry Pi, which is used to both send the Raspberry data concerning the movement of the different motors and to receive the video stream from the robot's onboard camera. The Raspberry then communicates over Serial port with the Arduino, which takes care of the data received to obey the user's commands.

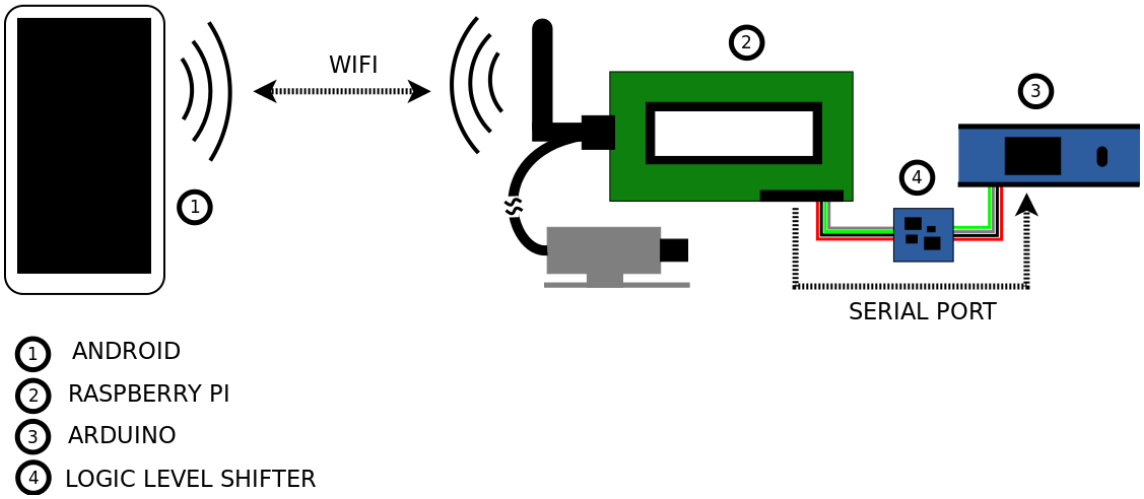

Figure 15: Logic connections diagram

8 Arduino

8.1 Overview

The Arduino is designed to control a relatively high number of devices. While it is a standard code has the following funct() bla bla...

8.2 Code

In this section the code written for the robot's controller will be explained. A flowchart diagram of the program is illustrated by Figure 16.

As it can be seen, the Arduino first defines all the robot's data, including the motor, servomotor and communication pins. This ensures the microcontroller knows where to send each datum once the user connects to the robot.

The program then enters its Loop function. Here it will check if the serial port is available, eg the user has sent a stream of data. Once the port is available, the Read function is called, which stores every byte received into a string to be used later. Once the reading has ended the code checks if it has received a special end-of-line character that signals the end of the data stream. If all the data was retrieved the code moves on to the next function.

The Parse function is called upon next. This function's purpose is to break and convert the previously stored string into the corresponding variables needed by each element, taking into account their sizes and types. Hence, it transforms one line of numbers into many parameters such as rotation angle, arm selection or movement direction which will be used by the next function.

With the data correctly formatted, the program executes the Process function which is where the "thinking" is done. Here are defined all the rules the robot must follow, such as knowing which claw to close depending on the side chosen by the user but closing both if the symmetry box was checked. It takes the data provided by the previous function and processes them to end up with a structured list of variables ready to be assigned to each element.

In the next step the Write function is called. This very simple function goes through the previous list assigning each variable to the corresponding element's assigned pins.

Finally, the code clears the initial string to make space for new data, resets the flag informing of the correct retrieval from the serial port and proceeds to the next iteration within the Loop function, restarting the process.

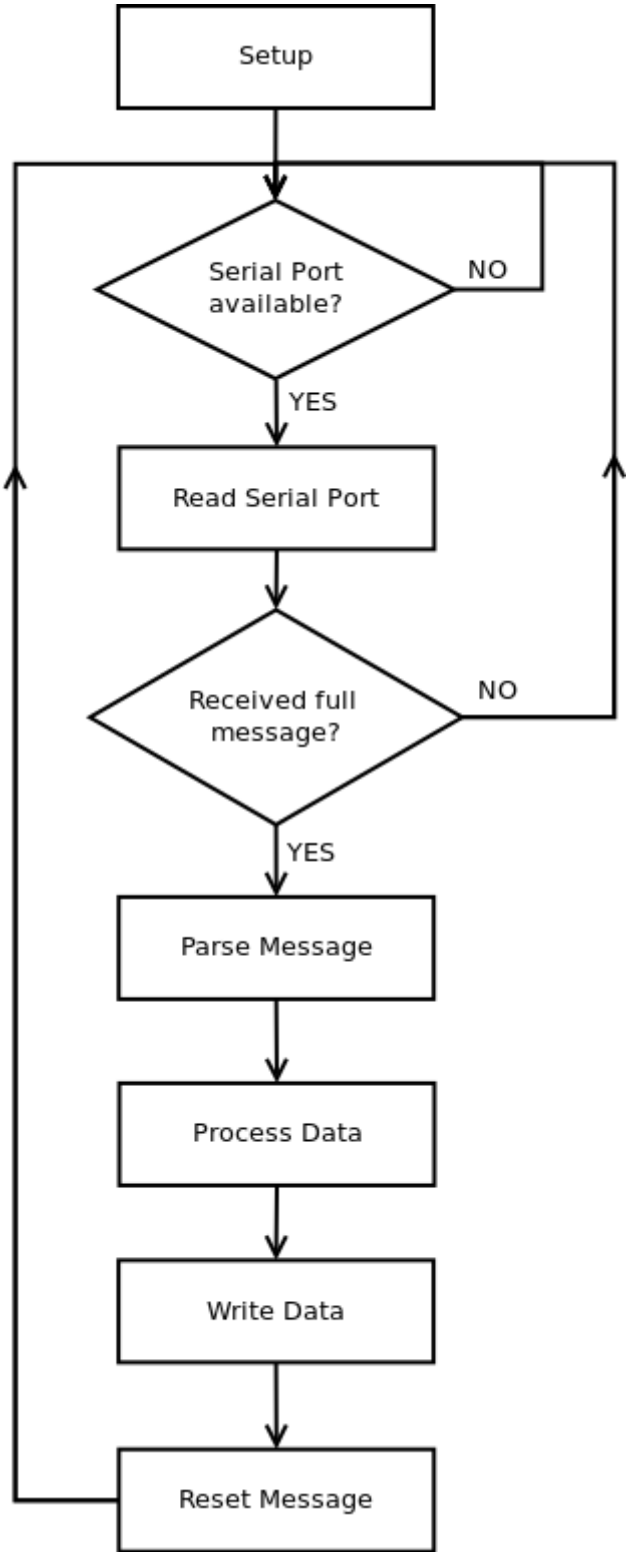

Figure 16: Arduino program flowchart

9 Raspberry Pi

The Raspberry Pi carries out three main duties to ensure everything works correctly. These include creating a wireless connection, streaming images from the camera to the phone and transmitting the data received from the phone to the microcontroller.

These are all placed into the */etc/rc.local* file so the system initializes them automatically each time the robot is turned on, with no need for human interaction.

9.1 Wireless Communications

The chosen means of communication between human and humanoid was wifi. This is so because it is a widely established technology, with great compatibility and in a great number of cases is already installed in the homes of users. It also has a greater speed and range than other technologies, like bluetooth, which are an asset in the case of streaming images.

Three methods were considered: connection to an existing wifi network, creation of an Ad-Hoc connection and establishment of a wifi Access Point.

9.1.1 Existing network:

The most straightforward solution is to simply connect the robot to the user's existing wifi network. This enables the user to control it from anywhere in the world, expanding its uses. However, some configuration is required, namely selecting the desired network and introducing the password, which complicates the setup by having to add a keyboard and a display.

This method would thus be suitable for experienced users and developpers, but not necessarily for the average seniors it is intended to help.

9.1.2 Ad-Hoc connection:

The next solution implemented was an Ad-Hoc connection between the Raspberry Pi and the Android phone.

This configuration aimed to solve the problem of usability, since the phone would automatically connect to the network, hence eradicating the problem of setting up the communication. This also had the advantage of creating an independent network, and thus being able to operate in remote areas.

In order to create implement this two files need to be set up. Firstly, the computer must be given the specific details of the new network to be created. Here, the contents of Listing 1 must be included into the file */etc/network/interfaces* .

Listing 1: Ad-Hoc Configuration [/etc/network/interfaces]

```
auto lo
iface lo inet loopback
iface eth0 inet dhcp

auto wlan0
iface wlan0 inet static
  address 192.168.1.1
  netmask 255.255.255.0
  wireless-channel 1
  wireless-essid RPiAdHocNetwork
  wireless-mode ad-hoc
```

With this configuration the Raspberry will assign itself the IP address 192.168.1.1, but the client computer will be left without an IP assigned and so will be unable to connect to the former.
To provide an IP to the client the package *DHCP Server* must be installed by typing *sudo apt-get install dhcp3-server* into a terminal.

Listing 2 must then be included in file */etc/dhcp/dhcpd.conf*

Listing 2: DHCP Server Configuration [/etc/dhcp/dhcpd.conf]

```
ddns-update-style interim;
default-lease-time 600;
max-lease-time 7200;
authoritative;
log-facility local7;
subnet 192.168.1.0 netmask 255.255.255.0 {
                range 192.168.1.5 192.168.1.150;
}
```

After rebooting the Raspberry Pi, the Ad-Hoc network is created and ready to use. However, while it is compatible with a large range of devices like computers and iOs devices, non-rooted Android devices are not able to connect to the network, which invalidates this procedure as it is not suited for the general public.

9.1.3 Wifi Access Point:

The last option for connecting the phone to the robot is to create a wifi Access Point (AP). This method involves a more complex setup than the previous, but while maintaining the same benefits, both allows Android devices to connect and supports speeds of up to 54 Mbps in 802.11g, while the former was limited to 11 Mbps in 802.11b.

In order to create the AP the *Hostapd* and *Isc-dhcp-server* packages have to be installed: *sudo apt-get install hostapd isc-dhcp-server*.

Once the needed packages have been installed the Dynamic Host Configuration Protocol (DHCP) server must be configured in order to assign IP addresses to clients. The contents of */etc/dhcp/dhcpd.conf* must be replaced with those in Listing 3.

Listing 3: DHCP Server Configuration [/etc/dhcp/dhcpd.conf]

```
# Sample configuration file for ISC dhcpd for Debian
# Attention: If /etc/ltsp/dhcpd.conf exists, that will be used as
# configuration file instead of this file.

# The ddns-updates-style parameter controls whether or not the server
# will attempt to do a DNS update when a lease is confirmed. We default
# to the behavior of the version 2 packages ('none', since DHCP v2
# didn't have support for DDNS.)
ddns-update-style none;

default-lease-time 600;
max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file
log-facility local7;

subnet 192.168.42.0 netmask 255.255.255.0 {
range 192.168.42.10 192.168.42.50;
option broadcast-address 192.168.42.255;
option routers 192.168.42.1;
default-lease-time 600;
max-lease-time 7200;
option domain-name "local";
option domain-name-servers 8.8.8.8, 8.8.4.4;
}
```

The next step is to establish the interface on which DHCP Server should assign IP addresses. This is done by copying the contents of Listing 4 to the file */etc/default/isc-dhcp-server*.

Listing 4: DHCP Server Defaults [/etc/default/isc-dhcp-server]

```
# Defaults for dhcp initscript
# sourced by /etc/init.d/dhcp
# installed at /etc/default/isc-dhcp-server by the maintainer scripts

#
# This is a POSIX shell fragment
#

# On what interfaces should the DHCP server (dhcpd) serve requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACES="wlan0"
```

Afterwards, the "wlan0" interface must be set up. In this case any previous configuration will be deleted by replacing the contents of */etc/network/interfaces* with those of Listing 5.

Listing 5: Interface Configuration [/etc/dnetwork/interfaces]

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow hotplug wlan0

iface wlan0 inet static
  address 192.168.42.1
  netmask 255.255.255.0
```

The DHCP configuration is now complete.

The Access Point setup has to be established next. A password-protected network will be created to ensure a secure connection. In this case its name will be "RaspiWifi" and its password "raspberry". Again, the contents of */etc/hostapd/hostapd.conf* should be replaced by those of Listing 6. This file is very sensitive, so no extra spaces are allowed.

Listing 6: AP Configuration [/etc/hostapd/hostapd.conf]

```
interface=wlan0
driver=rtl871xdrv
ssid=RaspiWifi
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Finally, the Raspberry has to be told where to find the configuration file previously created. The file */etc/default/hostapd* must include the contents of Listing 8.

Listing 7: AP Defaults [/etc/default/hostapd]

```
# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about
# alternative methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd
# configuration file and hostapd will be started during system boot.
# An example configurationfile can be found at
#/usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

The only thing remaining is to start the AP service at boot, which is done with the command *sudo update-rc.d hostapd enable*.

This concludes the wireless communications setup, finally using the wifi Access Point method because of the advantages mentioned previously.

9.2 MJPG Streamer

9.3 IP/UART Bridge

It uses the Adafruit-CharLCD library, which can be downloaded from their repository, to enable writing to the LCD screen.

Figure 17 presents a flowchart of the socket to serial connection software.

The program creates a TCP socket server which continuously searches for clients until one of them connects. Once a connection is secured, the LCD changes from "Awaiting clien" to

"Client connected" and the program waits instead to receive data from the client. The data received is examined to check if it is a "quit" string, in which case the connection is closed and the program awaits another client. On the other hand, if the data is a valid string from the client, it is passed on through serial communication to the Arduino for it to use.

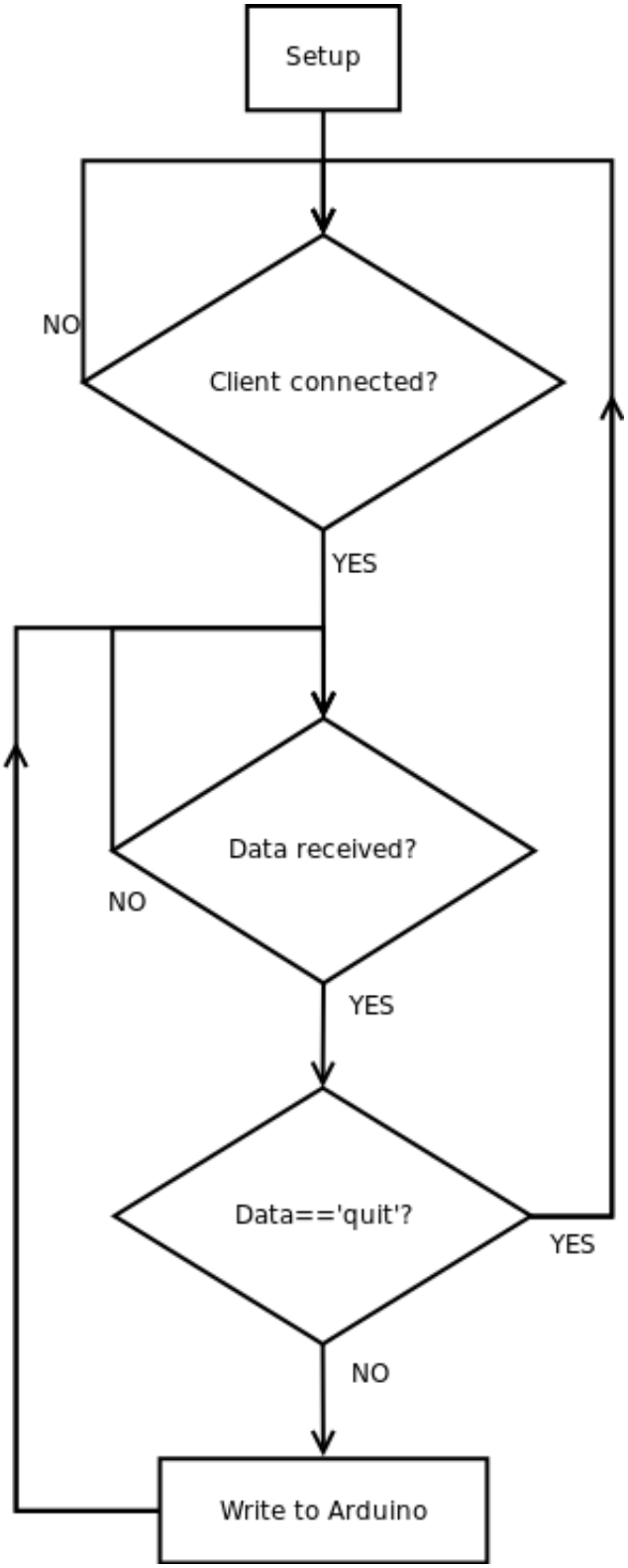

Figure 17: IP/UART program flowchart

9.4 Initializing Script

One of the defining features of this project is that it must be usable by non-technological people, and so it must initialize every service it needs on its own.

To do so, the tasks previously defined are called automatically from a script when the system boots. The contents of */etc/rc.local* are executed right after the computer executes its own routines.

Listing 8: Initialization Script [/etc/rc.local]

```
#!/bin/bash

#Start ip Server
/etc/init.d/isc-dhcp-server start

#Start webcam streaming
#Runs on background so this script is able to launch the next item
#in list
/home/pi/mjpg-streamer/startStreaming.sh &
sleep 0.3

#Start Android to Arduino dumping
/home/pi/AndroidToArduino/startA2A.sh


exit 0
```

The file must be given executable permissions in order to be allowed to implement the commands specified. This is done by typing *sudo chmod +x /etc/rc.local* into a terminal window.

10 Android

Android has been used instead of iphone or windows because ... marketshare, graphs, prices etc
app skeleton, typical app example, android studio etc
this app

Part IV
Conclusion

References

[1] EXAMPLE CITATION: Leslie Lamport, *LaTeX: a document preparation system*. Addison Wesley, Massachusetts, 2nd edition, 1994.