Jenny 5 - the robot

Mihai Oltean
Faculty of Exact Sciences and Engineering,
"1 Decembrie 1918” University of Alba Iulia,
Alba-Iulia, Romania.
mihai.oltean@gmail.com
https://jenny5.org
https://jenny5-robot.github.io

February 23, 2021

Abstract

Jenny 5 is a fully open-source robot intended to be used mainly for research but it can act as a human assistant too. It has a mobile platform with rubber tracks, a flexible leg, two arms with 7 degrees of freedom each and head with 2 degrees of freedom. The robot is actuated by 20 motors (DC, steppers and servos) and its state is read with the help of many sensors. The robot also has 3 webcams for computer vision tasks. In this paper the current state of the robot is described.
CONTENTS

1 Introduction 4

2 Source code 7

3 Specifications 8

4 Hardware design 9
   4.1 CAD software ........................................... 9
   4.2 CAD project structure .................................. 10
      4.2.1 Parameters .......................................... 10

5 Hardware 11
   5.1 Materials ................................................. 11
   5.2 Tools required to build Jenny 5 ........................... 11
   5.3 Mobile platform ......................................... 11
   5.4 Leg ....................................................... 12
   5.5 Arms ..................................................... 12
   5.6 Gripper .................................................. 15
   5.7 Body ..................................................... 15
   5.8 Head ..................................................... 18

6 Electronics 19
   6.1 Electrical / electronic materials ....................... 19
   6.2 Tools ..................................................... 19
   6.3 Arm custom PCB ......................................... 19

7 The brain 21

8 Power supply 21

9 The software controlling the robot 22
   9.1 Scufy - the Arduino firmware ............................ 22
      9.1.1 General commands .................................... 22
      9.1.2 Create controllers commands ......................... 22
      9.1.3 Attach sensors to motors .............................. 23
      9.1.4 Stepper Commands .................................... 23
      9.1.5 Servo commands ....................................... 24
      9.1.6 Read sensors commands ............................... 24
      9.1.7 Others ............................................... 25
   9.2 Scufy Lib - the PC library ............................... 25
      9.2.1 General commands .................................... 25
      9.2.2 Events processing .................................... 25
      9.2.3 Create commands ..................................... 26
      9.2.4 Stepper motor movements .............................. 27
      9.2.5 Attach sensors to motors .............................. 28
9.2.6 Reading Sensors .................................................. 28
9.2.7 State ................................................................. 29
9.2.8 Scufy Lib events ................................................... 29
9.2.9 Example of utilization ............................................. 29
9.3 RoboClaw control library ......................................... 31
9.4 HTML 5 client and PC WebSocket server ....................... 33
9.5 Intelligent algorithms ............................................... 33

10 Weaknesses and future development .............................. 36
1 Introduction

Jenny 5 represents an attempt to build a low-cost, almost humanoid robot, by using tools and materials which are readily available.

Jenny 5 is inspired by the Johnny 5 robot from the Short circuit movie [6].

Work to Jenny 5 robot was started in April 2015. Until now 3 major versions have been built and tested. At each iteration the robustness of the robot has been significantly improved.

Jenny 5 (v3) has a mobile platform with tracks, a pliable leg, two arms with 7 degrees of freedom each and one head. The current design of the Jenny 5 is given in Figure 1. A real world image is given in Figure 2.

All source files (CAD, software etc) for Jenny 5 are freely available on GitHub [2, 1]. All code is released under MIT license so that anyone can freely use it in both personal and commercial applications.

Several programming languages have been used for developing Jenny 5 hardware and software. These are: C++ (for server and the Arduino firmware), HTML5 and JavaScript (for the client controlling the server) and OpenSCAD (for the hardware design).

Jenny 5 is easy and cheap to build. Most components can be purchased from robotics stores. Some aluminum profiles can be cut and drilled with tools available for hobbyists. Custom made parts can be printed with a 3D printer.

Materials (excepting the onboard computer) cost less than 2500 USD.

The robot can be utilized in a wide range of scenarios. Here is a short list of things that the robot could do (if programmed properly): surveillance, rescue, disasters management, house cleaning, food preparation, cleaning kitchen table, working in the garden, fire fighting, combat missions etc.

The Jenny 5 robot is continuously developed and updated. This paper describes the current state of the robot (also called v3).

The document is structured as follows:
Section 2 contains the locations where the source code of the robot is stored.
Section 3 contains a list of technical specifications for Jenny 5.
Section 4 introduces OpenSCAD which is the software used to design the robot. Later the CAD project structure is given.
Section 5 describes the main components of the robot (mobile platform, leg, arms, body and head). A short list of materials and necessary tools is given in sections 5.1 and 5.2
Section 6 describes the electronic boards used for reading sensors and for moving motors.
Section 7 gives the specifications of the computer controlling the robot.
Batteries utilized in this project are listed in section 8
Section 9 describes the software that controls the robot (Scufy - the Arduino firmware, Scufy Lib - Arduino PC control library and the HTML 5 and WebSocket server). Several intelligent algorithms used by Jenny 5 are described in section 9.5.
Section 10 contains a short description of some future development directions.
Figure 1: Jenny 5 robot. Design view
Figure 2: Jenny 5 robot. Real world view of the current iteration (v3).
2 Source code

Jenny 5 is open source and the entire source code is stored as a GitHub organization at: https://github.com/jenny5-robot/. Individual repositories can be downloaded from the above-indicated address. The Bill of Materials (BOM) and the assembly manual are also available online.

The source code is released under MIT license [23] so that anyone can use it free or paid, open-source or closed-source, private or public projects. The only request is to mention the author(s).

Below is the list of repositories and their web addresses:

- **CAD files** - can be downloaded from [https://github.com/jenny5-robot/jenny5-cad](https://github.com/jenny5-robot/jenny5-cad). It requires [OpenSCAD](https://open.scad) to view.

- **Scufy** - the Arduino firmware - can be downloaded from [https://github.com/jenny5-robot/Scufy](https://github.com/jenny5-robot/Scufy). It is used by the arm and head electronics. It requires Arduino IDE [8] to run and upload to boards.

- **Scufy Lib** - Arduino control library - can be downloaded from [https://github.com/jenny5-robot/jenny5-control-module](https://github.com/jenny5-robot/jenny5-control-module). It is used for sending commands to Arduino firmware from a PC. It is written in C++ and requires a C++ compiler.

- **WebSocket server and HTML5 client** - can be downloaded from [https://github.com/jenny5-robot/jenny5-html5](https://github.com/jenny5-robot/jenny5-html5). It is used to control the robot from a browser. The server is a console application running on the robot PC.

- **custom PCBs** - can be downloaded from [https://github.com/jenny5-robot/jenny5-electronics](https://github.com/jenny5-robot/jenny5-electronics). They require Fritzing [20] to view or modify.

- **the assembly manual** - can be read from [https://jenny5.org/manual/](https://jenny5.org/manual/). Note that currently this is under construction and will not be completed until the robot is completed.

- The **Bill of Materials (BOM)** is stored as a Google Spreadsheet at: [https://docs.google.com/spreadsheets/d/1SwaC4woJEvUEqn66lIG9MLyuAD7tsWIQtvDkaCQT1s/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1SwaC4woJEvUEqn66lIG9MLyuAD7tsWIQtvDkaCQT1s/edit?usp=sharing).
3 Specifications

Technical specifications are given in Table 1. Please note that some specifications are not computed, but based on my experience with the robot.

| Parameter                                      | Value                                                                 |
|------------------------------------------------|------------------------------------------------------------------------|
| Total Weight                                   | 40kg                                                                   |
| Width                                          | 80cm (from left arm extremity to right arm extremity)                  |
| Total height                                   | 1.8m (from ground to head tip)                                        |
| Leg height                                      | 35cm (fully compressed) to 95cm (fully extended)                      |
| Arm length (from shoulder connection to gripper center) | 70 cm                                                               |
| Arm weight lift                                 | <1kg (when fully extended). Can more if a spring is used. See section [5,5] for a discussion. |
| Arm (time for a up-down move)                  | 11 seconds. See section [5,5] for a discussion.                       |
| Speed                                          | <3km/h. Can run faster if a motor with a lower gear reduction is used. |
| Leg (time to compress/extend)                  | 7 seconds.                                                            |
| Autonomy                                       | Several hours (depends on the capacity of the batteries).             |

Table 1: Technical specifications of the current iteration of Jenny 5. Please note that some specifications can be improved when using better components.
4 Hardware design

This section describes the software used to design Jenny 5 and the structure of the CAD project.

4.1 CAD software

Jenny 5 is designed in *OpenSCAD* [7]. I have chosen this environment because it is mainly intended for programmers and I (the author) am a programmer.

Parts in *OpenSCAD* are designed using computer instructions instead of using the mouse.

*OpenSCAD* IDE is very simple: one writes instructions in the left window, and the result (after pressing F5) are seen in the top-right window. In the bottom-right window the programmer can display various information by using *echo* instruction.

The language offers several primitives (*cube*, *cylinder*, *sphere* etc), several Boolean operations (*union*, *difference*, *intersection*) and several transformations (*translate*, *rotate*, *mirror*, etc). By using only these primitives and operations is possible to create very complex geometries.

The programmer can write everything in the main file, but usually the instructions creating a part are grouped within a *module*.

For instance a simple program which creates a nut is the following:

```plaintext
module nut(external_radius, internal_radius, thickness)
{
    difference()
    {
        cylinder(h = thickness, r = external_radius, $$fn = 6);
        cylinder(h = thickness, r = internal_radius, $$fn = 50);
    }
}
// now call the module
nut(external_radius = 4, internal_radius = 2, thickness = 3.2);
```

For visualizing a part the user should press F5 (*Preview*) or F6 (*Render*). One can use the mouse to navigate through the designed part. Left button drag is used for rotate, right button drag for move and mouse wheel for zoom.

Note that for visualizing the entire robot we will use *Preview* mode only because *Render* is too slow (can take several hours to render the entire robot). However, when we need the *stl* file for 3D printing, we will use *Render* mode for that particular part.

The user can uncomment each module of the project to show it on screen.
4.2 CAD project structure

CAD files for the robot can be downloaded from [https://github.com/jenny5-robot/jenny5-cad](https://github.com/jenny5-robot/jenny5-cad). There are 2 main folders there: basic.scad and robot.

- **basic.scad** - contains general components like screws, nuts, motors, sensors, pulleys, gears, bearings, etc. These components can be used in other projects too.

- **robot** - contains components specific to Jenny 5 robot. Each part of the robot has its own folder. Main file is called jenny5.scad. To view it just open it in OpenSCAD and press F5.

The robot folder contains 5 subfolders:

- **arm**. The main file is arm.scad.

- **base_platform**. The main file is base_platform.scad

- **body**. The main file is body.scad. Please note that arms are connected to body only in the main file of the project (jenny5.scad).

- **head**. The main file is head.scad.

- **leg**. The main file is leg.scad.

4.2.1 Parameters

The current position of the leg and arms is stored as numerical values (angles). These parameters are stored in the following files:

- **robot/arm/arm_params.scad** - here we have the angles for arms.

- **robot/leg/leg_params.scad** - here we have the angle for leg.

One can modify these parameters in order to simulate a new position for leg or arms.

One can change the color of the plastic parts from file basic.scad/material_colors.scad.
5 Hardware

This section describes the most important mechanical components of the robot which are the mobile platform (see section 5.3), the leg (see section 5.4), two arms (see section 5.5), a body (see section 5.7) and a head (see section 5.8).

5.1 Materials

In Table 2 the materials needed for the mechanical parts are listed. Please note that electrical / electronic components (including sensors, cameras, micro-controllers, etc) are not listed here, but the motors are. The quantities are not always exact. For more details the user is encouraged to read the online assembly manual from [1].

| Material                                           | Quantity                                      |
|----------------------------------------------------|-----------------------------------------------|
| M3, M4, M8, M12 screws                              | Many (of various lengths)                     |
| M3, M4, M6, M8, M12 nuts                            | Many                                          |
| Washers                                            | Many                                          |
| Aluminum rectangular tube                          | Several meters (various sizes)                |
| Aluminum sheets                                     | Many of various sizes and thicknesses(3mm, 5mm and 10mm) |
| Carbon fire tube                                    | Several meters                               |
| PLA (for 3D printers)                              | About 2 kgs                                   |
| Rubber tracks                                      | 2                                             |
| Radial bearings                                    | Many (of various types)                        |
| Linear motors (for leg)                            | 2                                             |
| DC motor(for platform)s                            | 2                                             |
| Stepper motors with planetary gearbox              | 12 (Nema 17 - for arms) + 2                   |
|                                                     | (Nema 11 - for head)                          |
| Servo motors (for grippers)                        | 2                                             |

Table 2: A short list of materials needed for building Jenny 5. Some lengths and quantities are approximate. Details are given in the assembly manual.

5.2 Tools required to build Jenny 5

Tools required to build Jenny 5 are given in Table 3.

5.3 Mobile platform

The platform is driven by 2 DC motors with planetary gearbox controlled by a 15 Amps RoboClaw board [14].

The platform has two rubber tracks.

Several components of the robot are placed on platform: batteries, motor controllers etc.
Table 3: Tools to make the mechanical part of Jenny 5.

| Tool                  | Use                                  | Comment                                                                 |
|-----------------------|--------------------------------------|-------------------------------------------------------------------------|
| Digital caliper       | For measuring various sizes          | A 150mm and a 300mm caliper are required.                              |
| Mitre Saw             | Cutting aluminum profiles            | I have a Bosch GCM 10 MX, but any other models should work. Make sure that it has a blade which cuts aluminum! |
| Column drilling machine | Drilling aluminum, wood parts      | I have a Optimum B16 model, but any other models should work.          |
| Hole saw              | Cutting holes in aluminum profiles   | 15-44 mm diameter. I have a Bosch Progressor with changing heads.      |
| Screw drivers         | screwing...                          | Various sizes and types (slot, Philips, hex, hex socket) are needed.    |
| Drill bits for metal  | Making various holes in aluminum    | 2.5 mm, 3.3 mm, 3.9 mm, 6 mm, 8 mm, 12 mm diameter.                    |
| Wrenches              | for tightening screws                | Various sizes are needed (mostly for M3, M4, M6, M8 and M12 nuts).     |

The platform is depicted in Figure 3. Currently, the platform is quite small and the robot can become unstable (can fall) if abruptly moved when the leg is fully extended. However, if the leg is compressed, the robot is much more stable. So, the idea is to make only moves with low acceleration when the leg is extended. In the near future we plan to add the possibility to move the body back and forth so that we can control the center of gravity easily.

### 5.4 Leg

The leg is driven by two, 750N, linear motors with 100mm stroke, controlled by a 5 Amps RoboClaw board. Note that the motors were disassembled in order to increase the useful stroke.

In Figure 4 we have a picture with the lateral and frontal leg design view. The leg can be compressed so that arms can grab things from ground.

### 5.5 Arms

Each arm has 6 stepper motors with various gearbox reductions (27:1 and 50:1). Joint position is read using a magnetic rotation sensor (AS 5147 [16]). Motors and sensors are controlled with a Pololu A-Star board [10].

The 50:1 motors are used for lifting the entire arm and for moving the elbow. Theoretically they are capable of a lot of force. According to the specifications [24] the motors (without gearbox) have 52N.cm torque. That multiplied with
Figure 3: Jenny 5 platform with rubber tracks
Figure 4: Jenny 5 leg. Lateral and frontal view.
50:1 reduction means 2600N.cm. The efficiency of gearboxes is 73%, so the useful torque is around 1900N.cm. The motor does not rotate the arm directly, but through the help of a 47:14 pulley reduction and this means that the final torque is around 6370 N.cm. At a distance of 70 cm (arm length) it means a 91 N.cm, which means that it can lift about 9kgs at 70cm. Note that this is a theoretical upper bound because the gearbox does not support more than 600N.cm before damage can occur [24], so the practical upper bound is below 3kg at 70 cm. Also, note that neither friction nor arm’s weight were taken into account here. However the arm weight can be canceled if a spring is used. One more aspect to consider is that the torque of stepper motor decreases at high speeds [25], so again the practical upper bound for torque is lower than the one computed before.

Still, in our experiments (see for instance movie at https://www.youtube.com/watch?v=Rc-ppA9-12I we have been able to lift more than 3.5kg at 40cm. This means a torque of at least 1400N.cm and no gearbox damage was observed in long term.

Another problem is that 50:1 gearboxes are difficult to back-drive. In the near future we plan to replace the 50:1 gearboxes with 27:1 ones because the maximum allowed torque before damage can occur is similar.

Regarding the speed of the 50:1 gearbox motors we have observed and made the following computations: The maximum speed that we were able to obtain with the motor (no gearboxes attached was 1500 steps/second. After that speed the motor just stops). Knowing that the motor has 1.8 degrees steps, it means that we can about 7.5 complete rotations in 1 second. After the 50:1 gearbox we can make 0.15 of a complete rotation in 1 second. After the external 47:14 reduction we make 0.044 of a complete rotation in 1 second, that is about 16 degrees in 1 second. Thus, a complete up-down move (180 degrees) of the arm will take about 11 seconds. We are currently investigating other stepper motor drivers to see if more steps per second are possible.

Each arm is controlled independently. The electronic boards are placed on each arm separately.

The arm is depicted in Figure 5.

5.6 Gripper

Each arm has attached a gripper which is moved by a servo motor. The gripper is connected to the same board as the rest of the arm. The gripper has attached a web-cam used for recognizing the objects in the front of it.

The gripper is depicted in Figure 6.

5.7 Body

The body is a carbon-fiber frame which has support for 2 arms and connectors for leg (at the bottom) and head (at the top).

Body is depicted in Figure 7.
Figure 5: Jenny 5 arm.

Figure 6: Jenny 5 gripper.
Figure 7: Jenny 5 body.
5.8 Head

The head has 2 degrees of freedom ensured by two Nema 11 stepper motors. Position of the motors is read by 2 AS5147 sensors.

The head has a web-cam for object detection and an ultrasonic sensor for distance measurement.

All head components, except the camera, are connected to an Arduino Nano board.
6 Electronics

This section describes the electronics components utilized by Jenny 5. First we list the materials and tools required to build the electrical part of the robot. The we describe the custom PCBs build for arms and head.

6.1 Electrical / electronic materials

A list of materials is given in Table 4. Note that motors were listed in Table 2 (see the hardware section 5).

| Component                  | Quantity | Comment                                           |
|----------------------------|----------|---------------------------------------------------|
| Arduino Nano               | 1        | for head                                          |
| RoboClaw 7Amps             | 1        | Controls the leg motors                           |
| RoboClaw 15Amps            | 1        | Controls the base platform motors                 |
| Pololu A-Star mini         | 2        | Controls the arms. We have chosen this board instead of Arduino Nano because A-Star has more pins. |
| Custom PCBs                | 4        | For arms and head. More custom PCBs for platform and legs are under development. |
| Pololu A4988 stepper driver| 15       | for driving motors on arms and head.              |
| AS5147 rotary encoder      | 14       | for reading the position of each arm and head articulation |
| Logitech C920 webcam        | 3        | for video capture. It is place on head and arms (for detecting the grabbed objects). |
| HC-SR04 ultrasonic wires   | 1        | for raw obstacles detection                       |
| resistors                  | Several  |                                                   |
| capacitors                 | Many     |                                                   |
| connectors                 | Many     |                                                   |

Table 4: Electrical materials required to build Jenny 5.

This section describes the electronic used to control the robot.

6.2 Tools

Tools requires to build the electrical parts for robot are given in Table 5.

6.3 Arm custom PCB

The PCB for arm is depicted in Figure 8. The electronics was designed using Fritzing [20].
Figure 8: Jenny 5 arm electronics. Design view (top) and actual implementation (bottom).
7 THE BRAIN

| Tool            | Use                                | Comment          |
|-----------------|------------------------------------|------------------|
| Digital multimeter | for measuring currents and voltages |                  |
| Soldering iron  | for soldering electrical components | A 20-30W iron is OK. |

Table 5: Tools required to build the electronics / electrical part of Jenny 5

7 The brain

The brain of the robot is a 13” laptop, with i7-6700 processor, 128GB SSD, and 4GB RAM and is placed on the platform. The laptop has 3 USB-A ports (more ports is better, otherwise some USB-hubs are needed).

The laptop sends commands to A-star, Arduino and RoboClaw boards and read images from webcams.

Obviously, any other laptop can be utilized, preferably small and powerful. A low-cost computer like Raspberry Pi [21] can be used too if the robot is not doing any high costs tasks like computer vision.

8 Power supply

The Jenny 5 robot is powered by multiple Li-Po rechargeable batteries. Currently a 16 Ah battery powers both arms and the head and a 10 Ah battery powers the platform and the leg. The laptop is powered by its own battery.
9 The software controlling the robot

The robot components are controlled by various programs. Arms and head run the Scufy - the Arduino firmware which accepts commands sent from PC with the help of Scufy Lib. The RoboClaw boards run a proprietary firmware and accepts commands sent by a C++ library running on a PC.

9.1 Scufy - the Arduino firmware

A-star and Arduino boards run a specially crafted firmware called Scufy (see the source code in reference [3]) which is able to control multiple motors and read a variety of sensors: buttons, AS5147 magnetic rotary encoders, HC-SR04 ultrasonic, potentiometers, infrared, TeraRanger One etc.

The firmware is asynchronous and does not block the board when performing long running operations (such as moving a motor to a new position (which can take sometimes several seconds) or reading an ultrasonic sensor).

Scufy firmware accepts commands sent through the serial port.

All commands are terminated by # character.

Almost all commands send a response to the serial port. The response is terminated by # character.

Commands related to motors and sensors required an object (controller) to be created internally for each type of motor or sensor. This controller stores how many sensors of that type we have and more information about the Arduino pins where that motor/sensor is connected. etc. So, before moving motors or reading sensors the user must create a controller for them. The motors / sensors inside a controller are indexed from 0.

The list of most important Scufy commands is given below. The user should read the entire / updated list of commands from the source code of the program.

9.1.1 General commands

- \(T\#\) - Test connection. - Outputs \(T\#\).
- \(V\#\) - Outputs version string (year.month.day.build_number). eg: \(V2019.05.10.0\#\).

9.1.2 Create controllers commands

- \(CS\ n\ dir_0\ step_0\ en_0\ dir_1\ step_1\ en_1\ ...\ dir_{n-1}\ step_{n-1}\ en_{n-1} \#\)
  Creates the stepper motors controller and set some of its parameters.
  \(n\) is the number of motors, \(dir_k, step_k, en_k\) are the Arduino pins which command the direction, step and the enable state.
  Outputs \(CS\#\) when done.
  Example: \(CS\ 3\ 5\ 4\ 12\ 7\ 6\ 12\ 9\ 8\ 12\#\)

- \(CV\ n\ pin_0\ pin_1\ ...\ pin_{n-1} \#\)
  Creates the servo motors controller and set its pins.
9 THE SOFTWARE CONTROLLING THE ROBOT

\( n \) is the number of motors, \( \text{pin}_k \) are Arduino pins where the motors are connected.
Outputs \( CV \# \) when done.

- \( CA n \text{pin}_0 \text{pin}_1 \ldots \text{pin}_{n-1} \#
\)
Creates the AS5147 controller.

\( n \) is the number of sensors, \( p_k \) are Arduino pins where the sensors are connected.
Outputs \( CA \# \) when done.
Example: \( CA 3 18 19 20 \#
\)

9.1.3 Attach sensors to motors

- \( ASx n P_y \text{end}_1 \text{end}_2 \text{home direction} A_k \text{end}_1 \text{end}_2 \text{home direction} \#
\)
Attach, to stepper motor \( x \), a list of \( n \) sensors (like Potentiometer \( y \), Button \( z \), AS5147 etc).
\( y \) is the sensor index in the list of sensors of that type.
\( \text{end}_1 \) and \( \text{end}_2 \) specify the sensor angular position guarding the motor movement.
\( \text{home} \) specifies the home position of the motor.
\( \text{direction} \) specifies if the increasing values for motor will also increase the values of the sensor.
Outputs \( ASx \# \) when done.
Example: \( AS0 1 40 280 320 300 1 \# \)

9.1.4 Stepper Commands

- \( SMx y \#
\)
Moves stepper motor \( x \) with \( y \) steps. If \( y \) is negative the motor runs in the opposite direction. The motor remains locked at the end of the movement.
The first motor has index 0.
Outputs \( SMx d \# \) when motor rotation is over. If the movement was complete, then \( d \) is 0, otherwise is the distance to go.
Example: \( SM1 100 \#
\)

- \( SHx \#
\)
Moves stepper motor \( x \) to the home position.
The first sensor in the list of sensors will establish the home position. The motor does nothing if no sensor is attached.
Outputs \( SHx \# \) when done.
9 THE SOFTWARE CONTROLLING THE ROBOT

- **SDx#**
  Disables stepper motor x.
  Outputs SDx# when done.

- **SLx#**
  Locks stepper motor x in the current position.
  Outputs SLx# when done.

- **SSx speed acceleration#**
  Sets the speed of stepper motor x to speed and the acceleration to acceleration.
  Outputs SSx# when done.

- **STx#**
  Stops motor x.
  Outputs STx# when done.

- **SGx position#**
  Moves stepper motor x to position sensor position. The first sensor in list will give the position.
  Outputs SMx d# when motor rotation is over. If the movement was complete, then d is 0, otherwise is the distance left to go.
  Example: SG1 100#

9.1.5 **Servo commands**

- **VMx position#**
  Moves servo motor x to position.
  Outputs VMx d# when done. If the move is completed d is 0, otherwise d is 1.

- **VHx#**
  Moves servo motor x to the home position. Home position is given by the first sensor in the list of attached sensors. Outputs VHx#.

9.1.6 **Read sensors commands**

- **RAx#**
  Read the value of AS5147 sensor.
  Outputs RAx angle#.
9 THE SOFTWARE CONTROLLING THE ROBOT

9.1.7 Others

- **E#**
  The firmware can output this string if there is something wrong with a given command.

- **Iinformation#**
  The firmware can output this string containing useful information about the progress of a command.

9.2 Scufy Lib - the PC library

*Scufy Lib*[^1] is a C++ library that sends commands (as strings) and reads results to / from the electronic boards powering the arms, head etc. Communication between on board computers and the electronic boards happens on a serial port.

The following methods are a part of the *Scufy Lib*. The user should read the entire list of commands from the source code of the library.

9.2.1 General commands

```cpp
// connects to given serial port
bool connect(const char* port, int baud_rate);

// test if the connection is open; For testing if the Arduino is alive please send_is_alive method and wait for IS_ALIVE_EVENT event
bool is_open(void);

// close serial connection
void close_connection(void);

// sends (to Arduino) a command (T#) for testing if the connection is alive
// when the Arduino will respond, the event will be added in the list
void send_is_alive(void);
```

9.2.2 Events processing

```cpp
// reads data from serial and updates the list of received events from Arduino
// this should be called frequently from the main loop of the program in order to read the data received from Arduino
```
bool update_commands_from_serial(void);

// search in the list of events for a particular event type
// it returns true if the event is found in list
// the first occurrence of the event is removed from the list
bool query_for_event(int event_type);

// search in the list of events for a particular event type
// it returns true if the event is found in list
// param1 parameter will be set to the information from the param1 member of the event
bool query_for_event(int event_type, int* param1);

// search in the list of events for a particular event type
// it returns true if the event is found in list
// param1 parameter will be set to the information from the param1 member of the event
// param2 parameter will be set to the information from the param2 member of the event
bool query_for_event(int event_type, int* param1, intptr_t* param2);

// search in the list of events for a particular event type
// returns true if the event type matches and if the param1 member is equal to the parameter given to this function
bool query_for_event(int event_type, int param1);

9.2.3 Create commands

// sends (to Arduino) a command for creating a stepper motor controller
// several arrays of pin indexes for direction, step and enable must be specified
// this method should be called once at the beginning of the program
// calling it multiple times is allowed, but this will only fragment the Arduino memory
void send_create_stepper_motors(int num_motors, int* dir_pins, int* step_pins, int* enable_pins);
THE SOFTWARE CONTROLLING THE ROBOT

// sends a command for creating a AS5147s controller
// this method should be called once at the beginning of
// calling it multiple times is allowed, but this will
// only fragment the Arduino memory
void send_create_as5147s(int num_as5147s, int* out_pins);

// sends a command for creating a Tera Ranger One
// controller
// this method should be called once
// only one sensor is permitted per Arduino board
void send_create_tera_ranger_one(void);

9.2.4 Stepper motor movements

// sends (to Arduino) a command for moving a stepper
// motor to home position
void send_go_home_stepper_motor(int motor_index);

// sends a command for moving a motor with a given number
// of steps
void send_move_stepper_motor(int motor_index, int
num_steps);

// sends a command for moving two motors
void send_move_stepper_motor2(int motor_index1, int
num_steps1, int motor_index2, int num_steps2);

// sends a command for moving three motors
void send_move_stepper_motor3(int motor_index1, int
num_steps1, int motor_index2, int num_steps2, int
motor_index3, int num_steps3);

// sends a command for moving four motors
void send_move_stepper_motor4(int motor_index1, int
num_steps1, int motor_index2, int num_steps2, int
motor_index3, int num_steps3, int motor_index4, int
num_steps4);

// sends a command for moving multiple motors
void send_move_stepper_motor_array(int num_motors, int*
motor_index, int *num_steps);

// sends a command for stopping a stepper motor
void send_stop_stepper_motor(int motor_index);

// sends a command for moving a motor to a new sensor position
void send_stepper_motor_goto_sensor_position(int motor_index, int sensor_position);

// sends a command for blocking a motor to current position
void send_lock_stepper_motor(int motor_index);

// sends a command for disabling a motor
void send_disable_stepper_motor(int motor_index);

// sends a command for setting the speed and acceleration of a given motor
void send_set_stepper_motor_speed_and_acceleration(int motor_index, int motor_speed, int motor_acceleration);

9.2.5 Attach sensors to motors

// sends (to Arduino) a command for attaching several sensors to a given motor
void send_attach_sensors_to_stepper_motor(int motor_index,
                                          int num_potentiometers, int *potentiometers_index,
                                          int* _low, int* _high, int* _home, int* _direction,
                                          int num_AS5147s, int* AS5147_index,
                                          int* AS5147_low, int* AS5147_high, int* AS5147_home, int* AS5147_direction,
                                          int num_infrared, int* infrared_index,
                                          int num_buttons, int* buttons_index, int* button_direction);

// sends a command for reading removing all attached sensors of a motor
void send_remove_attached_sensors_from_stepper_motor(int motor_index);

9.2.6 Reading Sensors

// sends (to Arduino) a command for reading a AS5147 position
void send_get_AS5147_position(int sensor_index);

9.2.7 State

// returns the state of a motor
int get_stepper_motor_state(int motor_index);

// sets the state of a motor
void set_stepper_motor_state(int motor_index, int new_state);

9.2.8 Scufy Lib events

Strings received from Scufy firmware are translated by update_commands_from_serial() to events which are stored into a queue. Each event has a particular type. A short list of event types is given in Table 6. For more events the user is encouraged to read the scufy_events.h file from the Scufy Lib repository.

| Event                             | Meaning                                                                 |
|----------------------------------|-------------------------------------------------------------------------|
| IS_ALIVE_EVENT                   | Received if the Scufy firmware responded to a T# command                |
| STEPPER_MOTORS_CONTROLLER_CREATED_EVENT | Received after the stepper motor controller has been created.          |
| ATTACH_SENSORS_EVENT             | Received after sensors have been attached to motors                     |
| AS5147_READ_EVENT                | Received after the sensor has been successfully read.                   |
| STEPPER_MOTOR_MOVE_DONE_EVENT    | Received after the motor has finished the requested move.              |

Table 6: Some of the events of Scufy Lib.

9.2.9 Example of utilization

In this section we give some example of utilization for the Scufy Lib.

After sending a command to Arduino firmware, the PC program should wait for an answer in an asynchronous way. Since the answer is not instantaneous the programmer should create a loop where it waits for an answer. The basic idea is the following:

- send a command,
- use update_commands_from_serial() to extract strings sent by firmware,
- use query_for_event() to determine if a particular event has been received.
Below is an example of code which creates a stepper controller. This code is actually used by the server to create a controller for the left arm of the robot [4].

```c
bool t_left_arm_controller ::
create_stepper_motors_controller(char* error_string)
{
    int left_arm_motors_dir_pins[6] = { 5, 7, 9, 11, 3, 1 };
    int left_arm_motors_step_pins[6] = { 4, 6, 8, 10, 2, 0 };
    int left_arm_motors_enable_pins[6] = { 12, 12, 12, 12, 12, 12 };
    arduino_controller.send_create_stepper_motors(6,
        left_arm_motors_dir_pins,
        left_arm_motors_step_pins,
        left_arm_motors_enable_pins);

    bool motors_controller_created = false;
    clock_t start_time = clock();
    while (1) {
        if (!arduino_controller.
            update_commands_from_serial())
            Sleep(5); // no new data from serial ... we make a little pause so that we don't kill the processor

        if (arduino_controller.query_for_event(
            STEPPER_MOTORS_CONTROLLER_CREATED_EVENT,
            0)) { // have we received the event from Serial ?
            motors_controller_created = true;
            break;
        }

        // measure the passed time
        clock_t end_time = clock();

        double wait_time = (double)(end_time -
            start_time) / CLOCKS_PER_SEC;
        // if more than 3 seconds then game over
        if (wait_time >
            NUM SECONDS TO WAIT FOR CONNECTION) {
            if (!motors_controller_created)
                printf(error_string, ""
```
Cannot create left arm's motors controller!
Game over!

return false;
}
}
return true;

//

9.3 RoboClaw control library

RoboClaw library [5] is C++ library which send commands, on a serial port, to the RoboClaw board.
Currently the platform and the leg are controlled by RoboClaw boards.
The following functions are included in library:

// returns the library version
// the caller must not delete the pointer
const char* get_library_version(void);

bool connect(const char* port, int baud_rate);
void close_connection(void);
bool is_open(void);

// Read the board temperature. // Value returned is in 10ths of degrees.
double get_board_temperature(void);

// Read the main battery voltage level connected to B+ and B- terminals
double get_main_battery_voltage(void);

// Read RoboClaw firmware version.
// Returns up to 48 bytes
// (depending on the RoboClaw model) and
// is terminated by a line feed character and a null character.
void get_firmware_version(char *firmware_version);

// Drive motor 1 forward.
// Valid data range is 0 - 127.
// A value of 127 = full speed forward,
// 64 = about half speed forward and 0 = full stop.
bool drive_forward_M1(unsigned char speed);
// Drive motor 2 forward.
// Valid data range is 0 - 127.
// A value of 127 = full speed forward, 64 = about half speed forward and 0 = full stop.
bool drive_forward_M2(unsigned char speed);
bool drive_backward_M1(unsigned char speed);
bool drive_backward_M2(unsigned char speed);

// Read the current draw from each motor in 10ma increments.
// The amps value is calculated by dividing the value by 100.
void get_motors_current_consumption(double &current_motor_1, double &current_motor_2);

// Read the current PWM output values for the motor channels.
// The values returned are +32767.
// The duty cycle percent is calculated by dividing the value by 327.67.
void read_motor_PWM(double &pwm_motor_1, double &pwm_motor_2);

// The duty value is signed and the range is:
// -32768 to +32767 (eg. +100 % duty).
// The acceleration value range is 0 to 655359
// (eg. maximum acceleration rate is -100 % to 100 % in 100ms).
bool drive_M1_with_signed_duty_and_acceleration(int16_t duty, uint32_t acceleration);

// The duty value is signed and the range is:
// -32768 to +32767 (eg. +100 % duty).
// The acceleration value range is 0 to 655359
// (eg. maximum acceleration rate is -100 % to 100 % in 100ms).
bool drive_M2_with_signed_duty_and_acceleration(int16_t duty, uint32_t acceleration);

// Set Motor 1 Maximum Current Limit.
// Current value is in 10ma units.
// To calculate multiply current limit by 100.
bool set_M1_max_current_limit(double c_max);

// Set Motor 1 Maximum Current Limit.
// Current value is in 10ma units.
// To calculate multiply current limit by 100.
bool set_M2_max_current_limit(double c_max);

9.4 HTML 5 client and PC WebSocket server

The robot can be manually controlled by an HTML5 application running within the browser of a smartphone. The HTML5 application connects to the server running on the robot. The server is the one that actually execute the commands (move motors, read sensors) etc.

The server is built on a top of a light WebSocket server (single source file) written by Eduard Šuica [17]. The server uses TLSe library [18] for the secured communication protocol.

The server requires a certificate to run. A sample certificate has been generated and stored in the certificates folder of the server. This will work with no problems on smartphones running Android. However, for iOS a new certificate must be generated. More details on how to do this for iOS can be found in reference [22].

The Jenny 5 web client allows to control one motor (as in the case of arms) or maximum two motors (as in the case of platform, leg and head) at a time. In such scenario the utilization of the application is very simple:

• the user presses a button (to select the motor that he wants to move),
• the tilts the smartphone,
• the client application read the gyroscope of the smartphone,
• the client application send the angle to the server on the robot,
• and the robot acts accordingly.

The client can request a picture from robot and then displays it in the browser window.

The web client also accepts voice commands. This feature is implemented by using Speech Recognition from HTML 5 [19].

An screenshot of the HTML5 client is depicted in Figure 9.

9.5 Intelligent algorithms

Currently there are two intelligent algorithms implemented on the robot:

• An algorithm which finds the closest face and moves the head’s motors in order to center it on the camera view. Only the head is involved in this operation.
Figure 9: Jenny 5 HTML 5 web client. Red circle indicates the tilt of the smartphone.
• An algorithm which follows the closest person. First it detects the closest face and then if the person is too close, the robot moves backward, otherwise it moves forward.

Both algorithm uses the OpenCV [12] for face detection.
10 Weaknesses and future development

Jenny 5 robot is under active development. During the development of the Jenny 5 robot we have observed several weaknesses that we plan to address in future iterations. Some short term ideas are listed in Table 7.

| Task                                      | Purpose                                           |
|-------------------------------------------|---------------------------------------------------|
| Rotatable body                            | Could reach farther objects without moving the base platform |
| Springs on upper arm-body articulation    | Will lift heavier weights                         |
| Two legs                                   | More stability and could climb stairs             |
| Intelligent algorithm for grabbing a bottle| More intelligence                                |
| Change speed for stepper motors during run| Finer control of the arms                         |
| LiDAR on platform and head                 | Better detection of obstacles                     |
| Visualizing the LiDAR on the client application | Can send the robot far away and see what it sees |
| Visualizing the camera (real time video) on the client application | Can send the robot far away and see what it sees |
| Arms electronics build around Arduino Mega | Could attach more sensors to arms                 |
| Custom made PCBs for leg and platform      | Can have own software and can attach more sensors to them |
| Carbon fiber sheets                        | To reduce the total weight of the robot           |
| Body movable back and forward              | To make the robot more stable on inclined plane   |
| Magnetic rotary sensors on leg             | To determine the exact position of leg            |
| Magnetic rotary sensors on platform gears  | To determine the true speed of the platform       |
| Gyroscope on platform                      | To determine if the robot will fall               |
Acknowledgement

The author likes to thank to the following of his students which have helped him while working to older prototypes of Jenny 5:

- 2017 students from Intelligent Robots class: Alexandru Donea, Daniela Oniţa, Flaviu Suciu, Tudor Samuila, Daniel Leah, Leontin Neamțu, Todor Uţ, Marius Penciu, Florin Jurj, Alin Simina, Mihaela Roşca.

- 2016 students from Intelligent Robots class: Pop Ioana, Mureşan Andreea, Kisari Andrei, Turtoi Cristian, Mateş Ciprian, Chereghi Adrian, Buciu- man Mircea, Bochiş Bogdan.

- 2015 students from Intelligent Robots class: Andra Ristei, Horea Mureşan, Baciu Iulian, Bica Ioana, Bonte Aurelian, Ilieş Daniel, Lonhard Cristian, Lungana Niculescu Alexandru Mihai, Marian Cristian, Sorban Timea, Tiperciuc Corvin.
References

[1] Jenny 5 website, https://jenny5.org or https://jenny5-robot.github.io, last accessed on 2019.06.10

[2] Source code for Jenny 5 robot on GitHub, https://github.com/jenny5-robot, last accessed on 2019.06.10

[3] Scufy - the Arduino firmware, https://github.com/jenny5-robot/Scufy, last accessed on 2019.06.10

[4] Scufy Lib, https://github.com/jenny5-robot/Scufy-Lib, last accessed on 2018.11.18

[5] RoboClaw control library, https://github.com/jenny5-robot/jenny5-control-module, last accessed on 2018.11.18

[6] Short circuit movie, https://www.imdb.com/title/tt0091949/, last accessed on 2017.04.15

[7] OpenSCAD, https://www.openscad.org last accessed on 2018.11.18

[8] Arduino, https://www.arduino.cc, last accessed on 2017.04.15

[9] Arduino Nano, https://www.arduino.cc/en/Guide/ArduinoNano, last access on 2018.11.10

[10] Pololu A-Star, https://pololu.com/product/3145, last accessed on 2018.11.10

[11] Terra Ranger One, https://teraranger.com/, last accessed on 2017.04.15

[12] OpenCV library, https://opencv.org, last accessed on 2017.04.15

[13] A4988 Stepper Motor Driver Carrier, https://pololu.com/product/1182, last accessed on 2018.11.10

[14] RoboClaw website, http://www.basicmicro.com/, last accessed on 2019.06.10

[15] Sharp GP2Y0D805Z0F senszor, https://pololu.com/product/1132, last accessed on 2018.11.10

[16] Rotary Position Sensor, Magnetic, https://digikei.com/product-detail/en/ams/AS5147-TS_EK_AB/AS5147-TS_EK_AB-ND/5452350, last accessed on 2018.11.10

[17] Eduard Šuica’s repositories on GitHub, https://github.com/eduardsui last accessed on 2019.06.20

[18] TLSe, https://github.com/eduardsui/tlse, last accessed on 2019.06.20
REFERENCES

[19] Speech Recognition in HTML5, https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API, last accessed on 2018.11.18

[20] Fritzing, electronics designed software, http://fritzing.org/home/, last accessed on 2018.11.19

[21] Raspberry Pi, https://www.raspberrypi.org/ last accessed on 2018.11.19

[22] Generating certificates for iOS, https://github.com/mattdesl/budo/blob/dcbc05866f583e172d6b46c898048436ab84d0e/docs/command-line-usage.md#ssl-on-ios, last accessed on 2018.11.19

[23] MIT license, https://opensource.org/licenses/MIT, last accessed on 2019.07.08

[24] Stepper Online website, https://www.omc-stepperonline.com last accessed on 2019.07.08

[25] Nick Johantgen, Torque vs speed of stepper motors, Oriental Motor USA Corporation, https://www.orientalmotor.com/stepper-motors/technology/speed-torque-curves-for-stepper-motors.html last accessed on 2019.07.08