5795, avenue De Gaspé, #126, 514 418-8384 www.sensorica.co

Automated tilt angle and orientation angle system for Queen's University's PV Measurements QE System

Milestone 2 Report - Design

This is a co-creative document.
If you contribute to this doc **make sure you respect** the Content rules

---

Report written by: Tammy Lea Meyer, Jim Anastasiou, Tiberius Brastaviceanu
In collaboration with SENSORICA affiliates: Daniel Brastaviceanu, Maria Frangos, John, Abran Khalid, Ahmed AKL, Thomas Vanesse.

This is a high level overview of the system and all its main components. This report informs the prototyping stage of the system, the third Milestone. Feedback is sought from the project initiator prior to the prototyping stage.

Last modified on July 28th, 2015

Table of contents

State Diagram
Flowchart

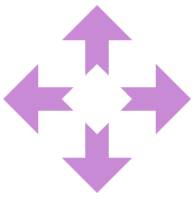
How to read this document

Growing consensus

The *Growing consensus* box is a summary of a section of this report.

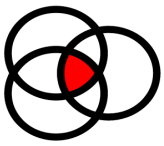
Attention

The *Attention* box points out some important things to consider.



Alternatives

The *Alternatives* box enumerates possible solutions to consider.



Reasoning

The *Reasoning* box presents arguments about possible choices.

Information

The *Information* box tells you how stuff works.

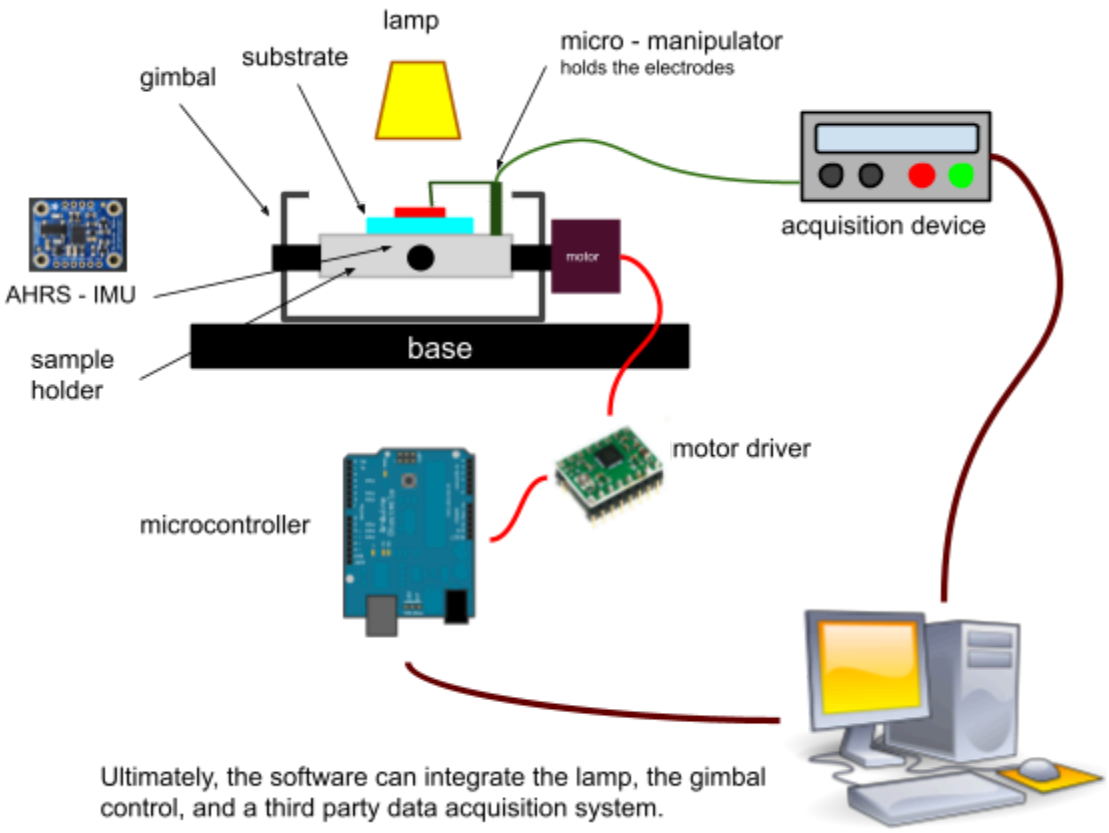
Project Description

Automated tilt angle and orientation angle system for Queen's University's PV Measurements QE System is a project in collaboration with Joshua M. Pearce, Department of Mechanical and Materials Engineering at Queen's University. This project is referenced within SENSORICA's network resource planning and value accounting system (NRP-VAS) as *PV Characterization*. Open project page.

Overview

The aim of the project is to design and create a system device for optical characterization of thin and transparent materials, as an open source project so it can have a maximum of social impact, and can be utilized and modified by communities around the world. The main application of the device is the measurement of the quantum efficiency (QE) of photovoltaic cells. The device comprises a moving stage from which photovoltaic filaments can be set to test the energy generated from any angle of incidence, within 1 degree of accuracy. There are some additional elements for fixing the sample and for electrical connections.

Main components and vocabulary



Milestone 1 mockup of system parts

- **Sample**: a flat photovoltaic (PV) cell; in our case one type of *Sample* would be a sheet of a fraction of a millimeter thick and approximately 2mm^2, mounted on a 1 square inch glass plate; another type of *Sample* would be a 3 mm thick and 6 inch square PV panel.
- **Sample substrate**: a flat material on which the *Sample* is fixed, in our case a glass plate of up to 3 mm thick, with a surface of a one square inch;
- **Sample holder**: a mechanical device used to hold the *Sample substrate* or the PV panel the in place. Also called **Sample plate** or **Stage**.
- **Gimbal**: an electromechanical assembly used to control the orientation of the *Sample* according to two axis of rotation; the axis of rotation must be contained within the plane of the *Sample*; the *Sample holder* is fixed on the *Gimbal*.
- **Motor**: electromechanical actuator used to change the orientation of the *Gimbal*
- **Motor driver**: electronic device used to control the *Motor*
- **Microcontroller**: an electronic board used to communicate with the *Motor driver*, in this case an Arduino board

- **Data acquisition device**: a third party device used to characterize *Sample*, to measure current, voltage, and perhaps other properties, as a function of light intensity, wavelength and direction of incidence
- **Electrodes**: thin wire (perhaps gold) used to make electrical connections between the PV Cell and the *Data acquisition device*
- **Manual micromanipulator**: a small format mechanical assembly used to position the electrodes on the PV cell with micron precision.
- **Lamp**: a light source simulating sunlight, used in the *Sample* characterization experiment
- **Base**: a rigid plate on which the entire device system is fixed.
- **Case**: the casing around the entire device system
- **Main axis**: the axis that determines the orientation of the entire device system, defined as the direction of propagation of the light from the *Lamp* to the *Sample*.
- **Micromanipulator**: is a mechanical device used to position and hold a thin wire on the contact surfaces of the PV cell, for electrical connections.

Open raw source [a working document used by those who collaborate on this project]

System requirements

General
The device should be low cost, open source, user friendly, modular, automatically log experimental data, and easy to repair.

Mechanical assembly
The mechanical assembly, described in general terms below, should be primarily 3D printable with a RepRap 3D printer. It should all fit into 1 cubic foot. It should be able to accommodate a *Sample* with a surface area from 1 square inch up to 6 square inches.

***Sample* orientation**
The device is used to control the *Sample* orientation in space, with the following parameters:

Accuracy	Precision	Repeatability
difference between the average and the real position (given by a "perfect" reference)	distance of every attempt to reach a position from average	closeness of agreement between the attained [positions] after *n* repeat visits to the same command pose in the same direction
0.5 to 1 degree	**0.5 degree**	**3%**

Control
The orientation of the *Sample* is automated. The hardware used will be controlled with open source software, running on Linux, and must be designed for easy interfacing with other open source software for systems integration and automation.

Physical dimensions limits
The entire device system should fit into a 1 ft cube.

Design considerations

Important considerations of the design of this device were discussed in Milestone 1 - Design considerations. Feel free to browse the Design considerations report.

Electronic Design

This section presents the design of the electronic layer of the device.

Parts selection

Brushless DC Motors

Brushless DC (BLDC) Electric Motor

BLDC motors have windings on the stator and permanent magnets on the rotor. Thus there is no need for brushes or commutators since there is no need to provide electricity at the rotor. By varying the current into the stator windings, a torque is developed via interaction with the rotor magnets. (Note that the term “BLDC” is actually a misnomer since although the current into the windings typically comes from a DC power source, the winding currents are actually AC in nature or there would be no resulting motive force.) BLDC motors eliminate many of the disadvantages of brushed DC motors, but do require more complex and costly electronics.

Motor options

We decided to go with brushless motors because they are higher torque, precise, and lighter than stepper motors. They are a bit more expensive than stepper motors, but not so much for equivalent torque.

Manufacturers:

Rc tiger motor, popular manufacturer of rc motors. Major worldwide distribution.
Turnigy power systems, seems to be owned by HobbyKing
iPower GBM Motors from iFlight, chinese manufacturer of drone motors
FoxtechFPV seems like chinese re-branding

Foxtech 5208-200T

Specification:
Configuration:12N14P
Wire: 0.24mm
Turn: 180 T
Motor size: Φ63*24
Shaft: hollow shaft/ Φ7.0 out, Φ5.0 inner
Ω Ri :17.2ohm
Weight: 180g
Foxtech

Turnigy HD 5206 Brushless Gimbal Motor

The 5206 is the perfect size for loads around 600-1500 grams.
Specs:
Poles: 12N14P
KV(RPM/V): 42
Resistance: 9.3omh
Weight: 150g
Wire: 0.27mm
Termination method: Star
Top holes center to center: 15mm
Bottom mount: M3 25 x 25mm
Sources:
HobbyKing

Turnigy HD 5208 Brushless Gimbal Motor

The 5208 is the perfect size for loads around 800-1900 grams.

Specs:
Poles: 12N14P
KV(RPM/V): 31
Resistance: 10 omh
Weight: 180g
Wire: 0.27mm
Termination method: Star
Top holes center to center: M2.5 15mm
Bottom mount: M3 25mm x 25mm

Sources: HobbyKing Amazon	
iPower GBM4108H-120T Weight:98g Motor Dimensions:45x25mm Stator Dimensions:41x8mm Copper wire(OD):0.17mm Configuration:12N14P Resistance:11.4 ohms Pre-wounded with 120 turns,hollow shaft Bottom mounting holes: see the draft Top mounting holes: 12mm center to center Camera range: 600-1200 grams **Sources:** iFlight	
GB4106 GB54-1 is even more powerful **specs to be added shortly...**	

Motor choice

The *Motors* above are widely used by RC communities. The GB4106 was highly recommended by RobotShop and another local Montreal suppliers of parts to the RC community. It is used in professional camera gimbals.

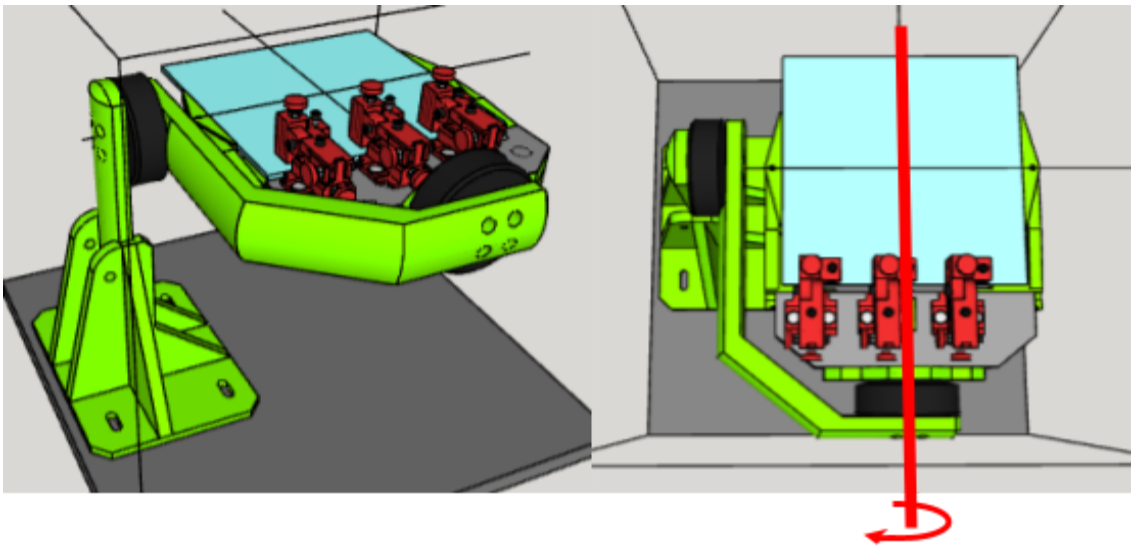
Rc tiger motor, popular manufacturer of rc motors. Major worldwide distribution.

Considerations about torque

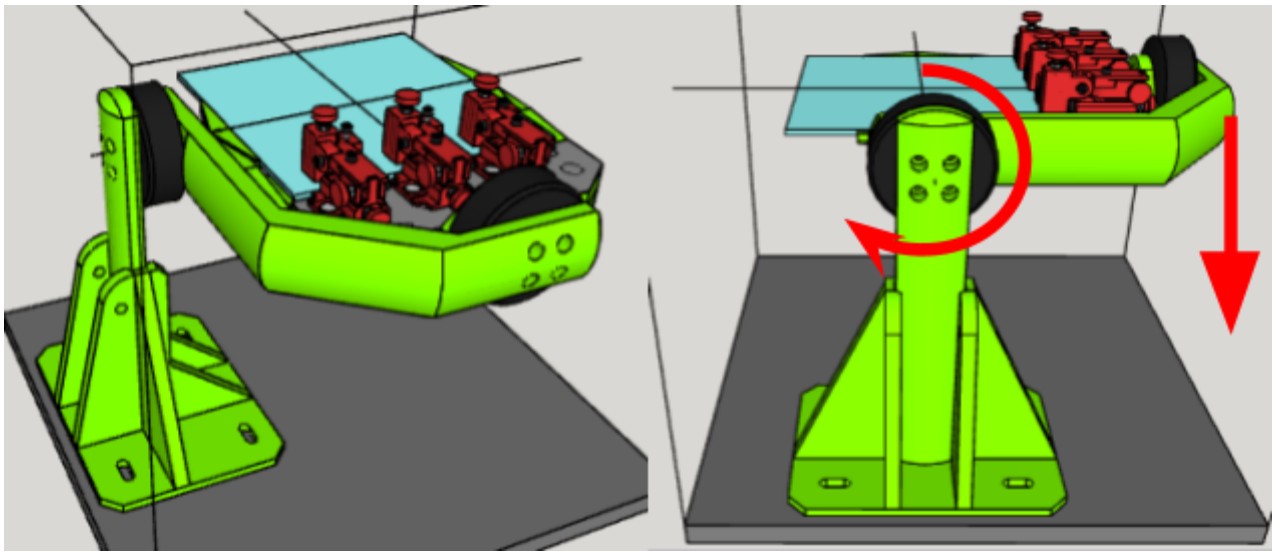
Two main parameters are important for choosing the stepper motor: accuracy of rotation (degree/step) and torque. The accuracy at the sample needs to be between 0.5 and 1 degree. The two mechanical designs proposed below have motion reduction features, and the most commun stepper motors are 200 steps/rotation, which comes to 1.8º/step (at full step), which can be further reduced if the motor is driven in microsteps to 0.9º (½ step) or 0.45º (¼ step). Therefore, accuracy in not a problem.

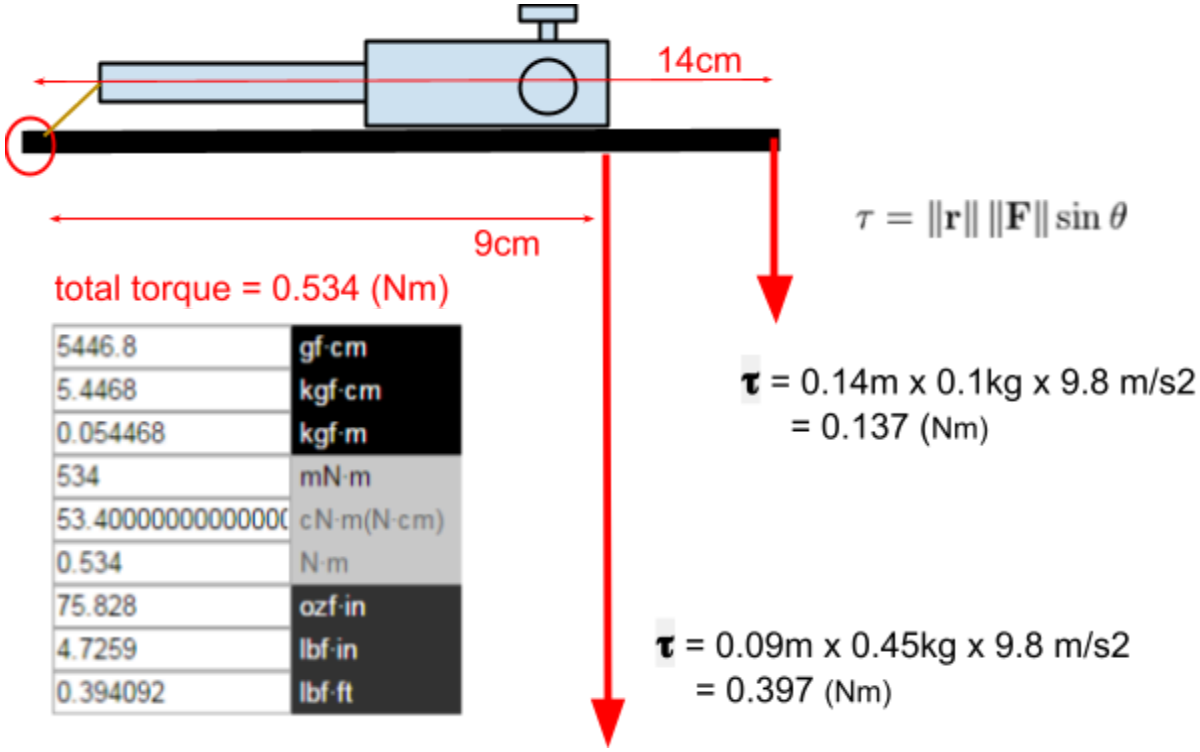
In order to estimate the required torque for the motors we need to make some assumptions. Since the requirements for speed and acceleration/deceleration are relaxed, we don't need a high torque to fight against the angular momentum of the gimbal. One concern is to have the gimbal steady in place, with the motors able to fighting against gravity if the weight on the gimbal is not evenly or symmetrically distributed.

The sample holder can be pretty well balanced along the secondary axis of rotation.



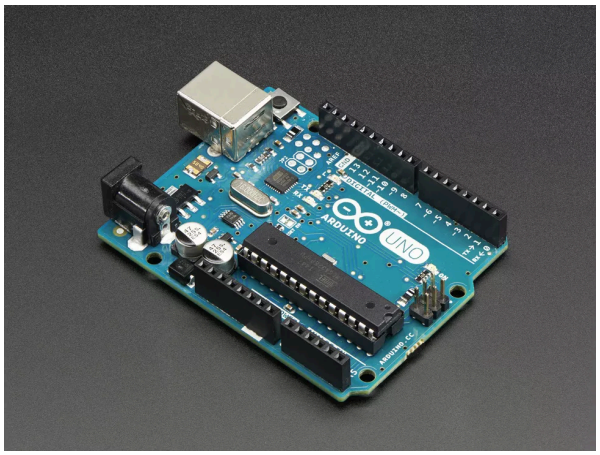
The problem arises for the *Motor* used to actuate the rotation around the main axis. It must support another motor and the 3 micromanipulators. The chosen brushless *Motor* weighs 100g, and it is placed at a distance of approx 14cm from the main axis of rotation. The center of gravity of the micronamipulators is approximately placed at 9cm from the main axis of rotation, each micromanipulator weighing approx 150g.





5446.8	gf·cm
5.4468	kgf·cm
0.054468	kgf·m
534	mN·m
53.400000000000(	cN·m(N·cm)
0.534	N·m
75.828	ozf·in
4.7259	lbf·in
0.394092	lbf·ft

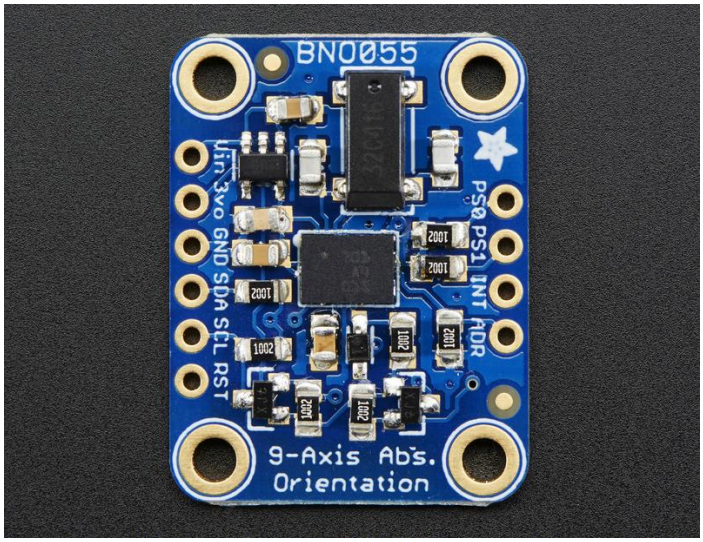
Motor controller



Arduino UNO R3 is one of the most famous boards in Arduino family and has the highest compatibility with shields and sketches.
Features:

- 6 analog inputs
- 14 digital I/O (6 PWM)
- 16Mhz clock speed
- ATmega328 processor with 32kB memory
- Dimension of 68.6mm x 53.3mm

Closed Loop Control System



Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055
Dimensions: 20mm x 27mm x 4mm / 0.8" x 1.1" x 0.2"

This board from Adafruit uses Bosch's **BNO055** SIP and combines the sensors and an ARM Cortex-M0 processors on a single die with the sensor fusion built in! Basically you don't need an external microcontroller running the libraries that turn the raw data into usable output.
PDF, Guide

Controler and closed loop system in one

Recommended to study this open hardware driver board based on L6234 BLDC Motor Driver by Michael Anton. It has input protection resistors, zeners, power supply/filtering components and even back-EMF sensing circuit with amplifier (not used here).
L6234 Datasheet (not very useful) go to application note AN1088 instead for use case.

Board options:

- RCTIMER 2-Axis Brushless Gimbal Controller & IMU v1.0;
- BaseCam (AlexMos) SimpleBGC (don't know where to buy);
- V3 Brushless Gimbal Controller - 2 Axis (Genuine Martinez), or from RC man child

Martinez V3 board - wiring diagram
See how to Martinez Brushless Gimbal, very promising and well-documented.
All the code for the Arduino and the board is HERE. [Tibi and Ahmed looked at the code for the Martinez 2 axis motor driver, in the BruGi_050_r200.zip and we understood high level, how it is possible to disable the code for the driver's own IMU and insert code for the 9DOF, at the Arduino level, and how to insert code for the Raspberry PI.]

Another lower cost alternative: Universal 2-axis Brushless Gimbal Controller Open Source V046 with Sensor, or see the same on Aliexpress, and on RCDroneHobby.

Driving a three phase BLDC motor with an Arduino

Great starting point
Even better one

There is some code here for six step sinusoidal commutation using lookup tables which I'd like to test. My main concerns are back EMF and noise introduced in the I2C signal from the IMU.

It turns out the most common design for driving a BLDC is to use an ESC controlled by a PWM signal from the Arduino. TODO: find the most appropriate ESC chip for this project
One very important thing to keep in mind when playing with the ESC is that it can't be supplied with the same source as the Arduino (what a surprise...) and one should **never** connect the ESC+its power supply to the Arduino when another power source is connected at the same time (such as the USB port for instance). So this leads us with the need for a DC power supply that will power the entire system. TODO: find the most appropriate power supply for this project

That said, setting up 2 ESCs with our BLDCs and the Arduino is pretty straightforward as this guy describes it.

Power supply

NOTE: Missing section. We'll choose one as soon as we'll have the mechanical design fixed, which will also determine the size of the stepper motor.

Alimentation:

Moteurs = puissance (V et A) et nombre(3) ? le driver (BLDC Motor Driver by Michael Anton) des moteurs peut fournir 5A peak par sortie, donc alimentation moteur doit pouvoir fournir au moins 5A peak. Disons qu'il fonctionne a 12V...

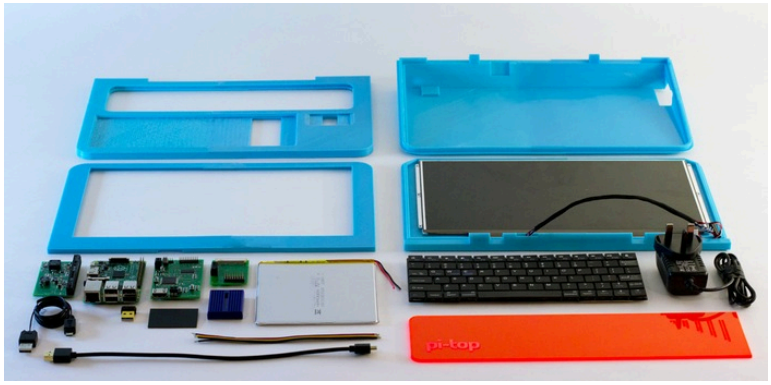
Arduino et composants = moins de 5V/700mA
Laptop (Raspberry 2A + composant + écran) = 57.72 watt-heure soit 12V/5A
Marge de sécurité = 1A
Alimentation doit être environs 30A
Je recommanderais toujours une alimentation par commutation

exemple:
http://www.amazon.com/Universal-Regulated-Switching-Computer-Project/dp/B00D7CWSCG
http://www.ebay.ca/itm/12V-DC-30A-360W-New-Regulated-Switching-Power-Supply-for-LED-Strip-Light-Metal-/131488243781?hash=item1e9d4f6045
http://fr.aliexpress.com/item/Convert-AC-110V-260V-to-DC-12V-Switch-Power-Supply-30A-360W-Voltage-Lighting-Transformer-for/32313241508.html

Pour avoir du 5V avec du 12V:
http://www.selectronic.fr/regulateur-de-tension-positive-5v-7805.html

Laptop Computer

Based on our assumptions that the requested laptop will only be used to run the PV application software we propose a Raspberry PI 2 based laptop, with a customized 3D printable casing, inspired by Pi Top. We estimated the cost to build it would be under 350$ with a 1920x1080 screen.



An attractive solution in alignment with the philosophy of Open Source Hardware would be to be to encourage the creator of Novena, the world's first open source Laptop that raised $782,505 during a crowdfunding campaign this May. Unfortunately it is priced between 1500$ and 5000$ for the wood version.

Wire management

Wire connections with the Data acquisition

One sensitive part is the data acquisition wires. They need to electrically connect to the gold wire, which makes the contact with the PV cell sample. Problems may arise when the sample is turned. There might be tension on the wire leading to the data acquisition device. This tension can transfer to the gold wire, which can affect the critical electrical connection. The larger wire that goes to the data acquisition device can be fixed on the *Sample holder* first, after to the *micromanipulators*, and in the end, to make an electrical connection with the gold wire. Special wire clamps will be designed for wire management.

Wire connection with the Feedback system

The ADHS-IMU feedback system needs to be connected to the Arduino board. This electronic board is placed directly on the *Sample holder*. The Arduino is not placed on the gimbal. This wire connection is not sensitive, but we need to manage this wire properly in order to avoid shadows and twisting around, due to the rotation of the gimbal. Wire channels will be designed within the structure of the gimbal. Special wire clamps will also be designed for wire management.

Mechanical Design

This section explains the mechanical design of the entire device.

Gimbal

Our work lead us to two different designs: a *Spherical drive* and the *Worm gear drive.*

The *Spherical drive* design is an elegant three *Motor* mechanical solution that allows the control of the orientation of the *Sample* in 3 axes: rotation, tilt and tip. The electronic and software control systems are more complex for this solution, since active feedback and spherical coordinates need to be used.

The *Worm gear drive* design allows the control of the orientation of the *Sample* in two axis: tilt and tip. It uses two motors and the axes are uncoupled, which makes the electronics and software control systems straightforward.

Simple 2 axes rotation design

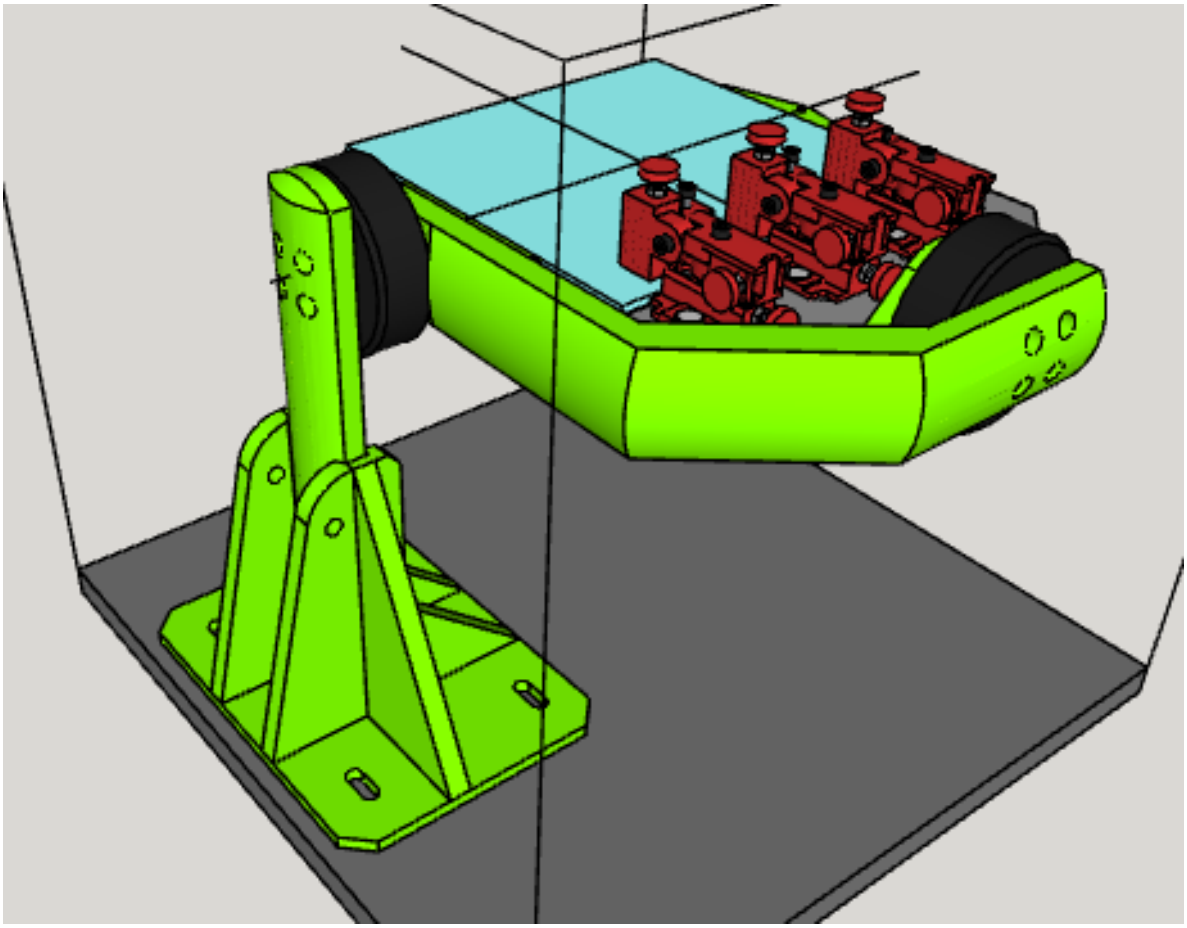
This design uses two motors to rotate the *Sample* around two axes, both of them contained in the plane of the sample *Sample*.

Link to 3D morel

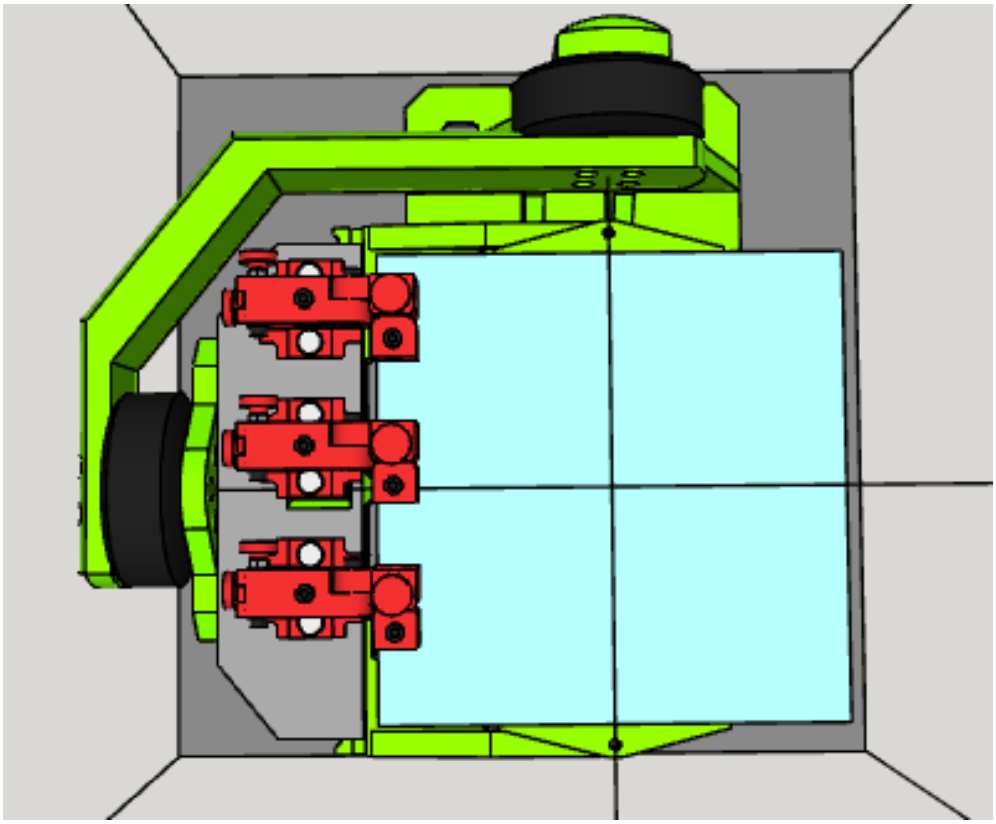
One axes, let's call it the main, is fixed with reference to the lab, and can perform a tip rotation of 360º. The secondary axis perpendicular to the principal axis within the plane of the sample, it can also rotate 360º.

Both motors transfer their rotational motion directly to the *Sample holder*, with no gears or other additional mechanisms. This design has been optimized to be used with brushless motors, which provide high precision and high torque.

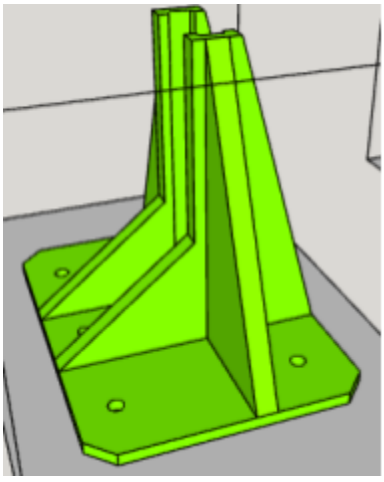
Side view. Also in the picture, 3 micromanipulators (in red)
the Sample clamp (in grey) and the Sample substrate (in blue).

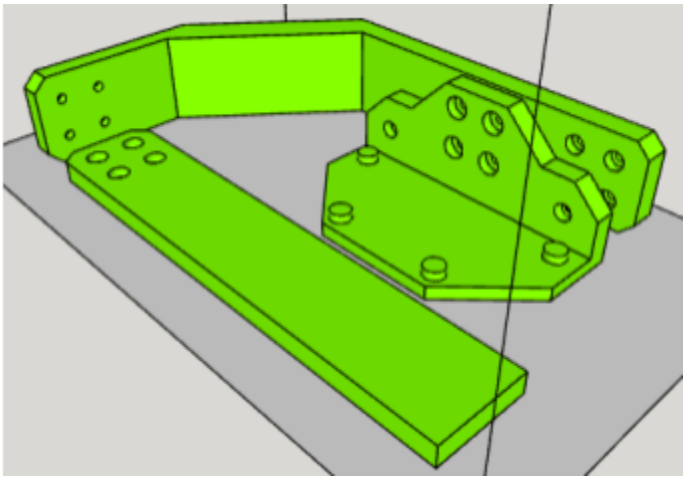


Top view of the gimbal. Also in the picture, 3 micromanipulators (in red) the Sample clamp (in grey) and the Sample substrate (in blue).



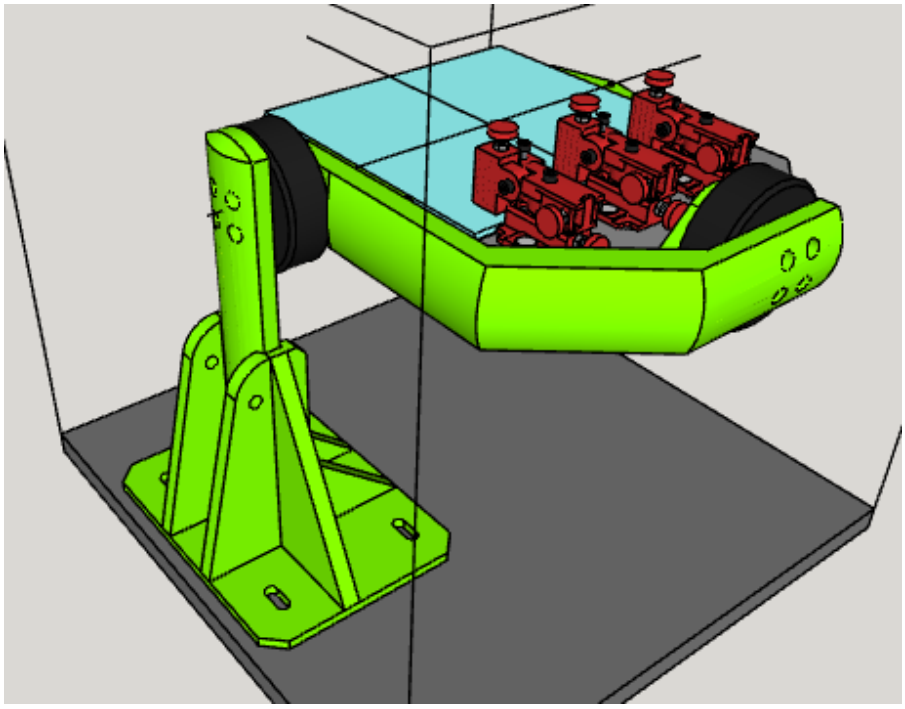
Individual components





Gimbal design choice

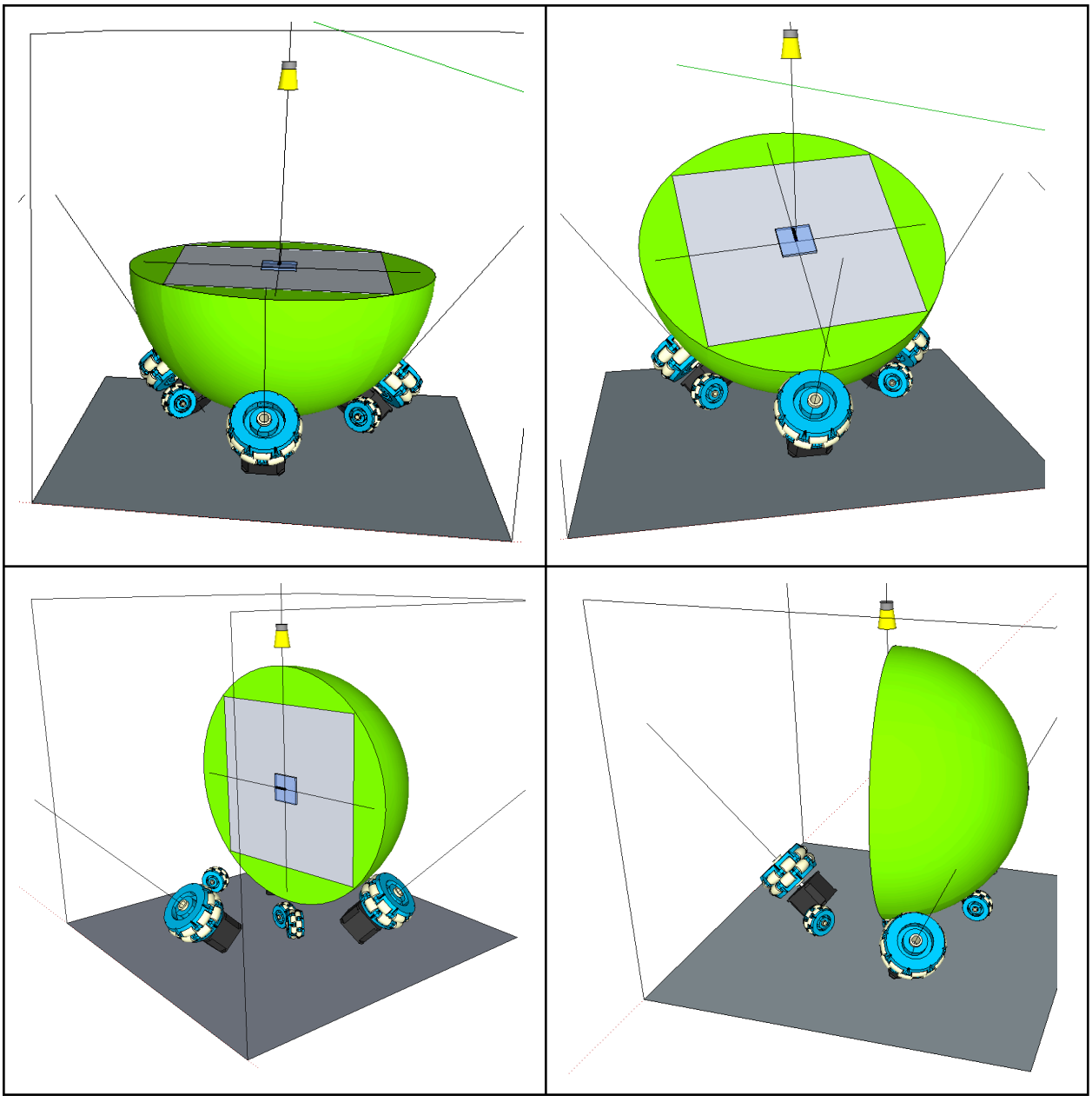
The simple 2 axis rotation design was chosen during a meeting with the project initiator. The merits of this design is its simplicity for electronic and software control, 3D printing of parts and assembly.

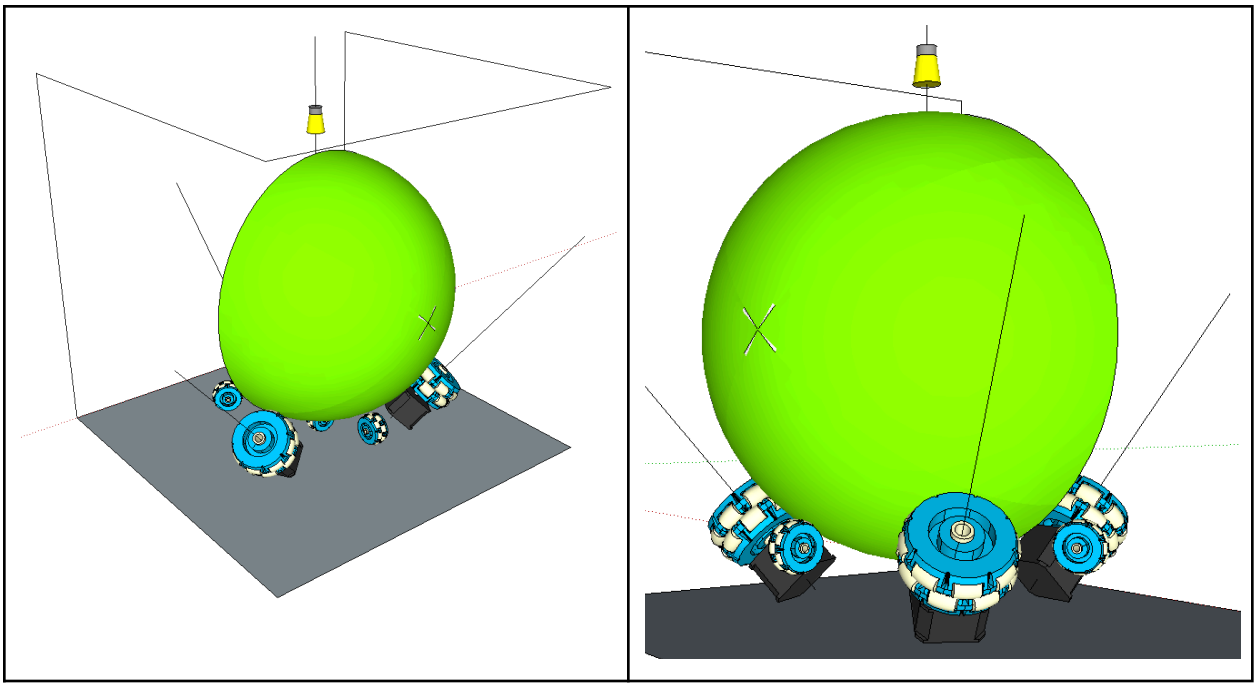


Spherical drive design

This device is made of a hemispherical body manipulated by three omni wheels driven by three *Motors*.

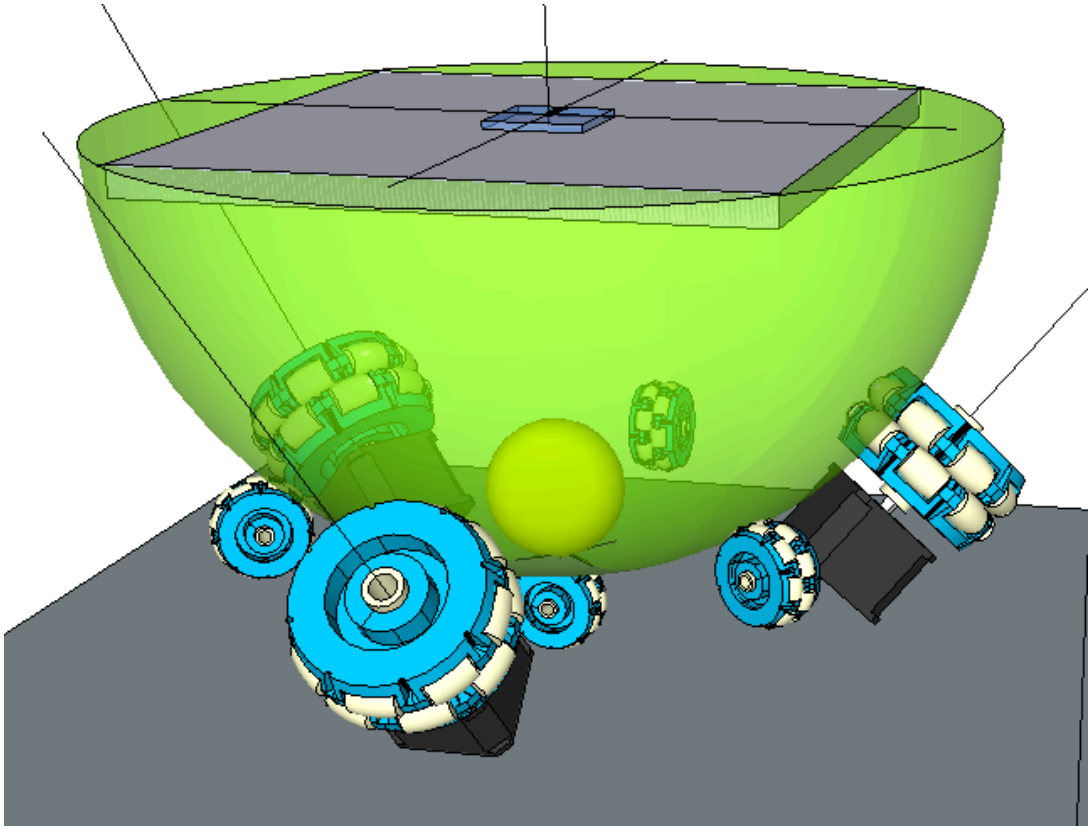
Link to 3D model



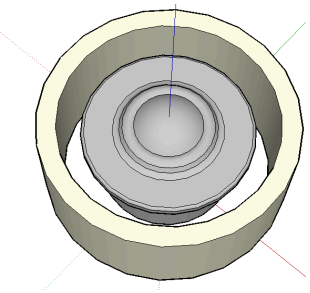


See [this video of an example of a similar technological solution in another application.]

In order to give the hemispherical body gravity and stability, a heavy metal ball is placed inside, which rolls to the lowest point. A strong magnet can also be placed beneath the hemispherical body to reduce bouncing of the heavy metal ball caused by friction and its own inertia.



The model shows four smaller omni wheels. Their role is to support the hemispherical stage. They can be replaced with ball transfers. At the bottom, the transfer ball can be made of a nonmagnetic material and surrounded by a hollow cylindrical magnet. The magnet is needed to pull the heavy metal ball within the hemispherical body close to the lowest point, and to prevent it from wobbling inside.



The hemispherical body is 3D printed. It is designed in small sections that can be handled by small size 3D printers, and can be easily assembled. In order to achieve the required precision of the motion, the surface of the hemispherical body may require some smoothing as post-processing. Moreover, in order to increase the traction of the omniwheels, the surface of the hemispherical body might need to be spray painted with a thin coat of adherent material. Inside the body, sanding or otherwise treating the interior of the hemispherical body might be required in order to allow the heavy metal ball to roll smoothly.

The electronic and software control system of this device is a little bit more complex than the *Worm gear* design. Moreover, this design uses three motors and the *Worm gear* design only uses two.

The main advantage of the of the *Spherical drive* design is that it allows all the possible orientations, eliminating the need to manually reorient the *Sample* in order to avoid shadows from other instruments on the *Sample holder* (for example the *Micromanipulators*). The *worm gear* design is more limited in that regard.

If this design is chosen, we will need to expand on the design elements in the prototyping stage.

This design was explored conceptually and the Arduino firmware was not designed for it. We estimate a few days of work to redesign the Arduino firmware for this three *Motor Spherical drive* design.

References

- Applied by Spherical Drive System to transportation
- *A Slip Model for the Spherical Actuation of the Atlas Motion Platform*; J.B. HOLLAND, M.J.D. HAYES, and R.G. LANGLOIS
- *A Complete Kinematic and Dynamic Model for Spherical Actuation of the Atlas Motion Platform*; M. Swartz, M.J.D. Hayes, R.G. Langlois

- *Velocity-Level Kinematics of the Atlas Spherical Orienting Device Using Omni-Wheels*; J.D. ROBINSON, J.B. HOLLAND, M.J.D. HAYES, R.G. LANGLOIS
- *Dynamic balancing mobile robot* Publication number US20080084175 A1

How to make a Ball Balancing Robot - Instructables

Worm gear design

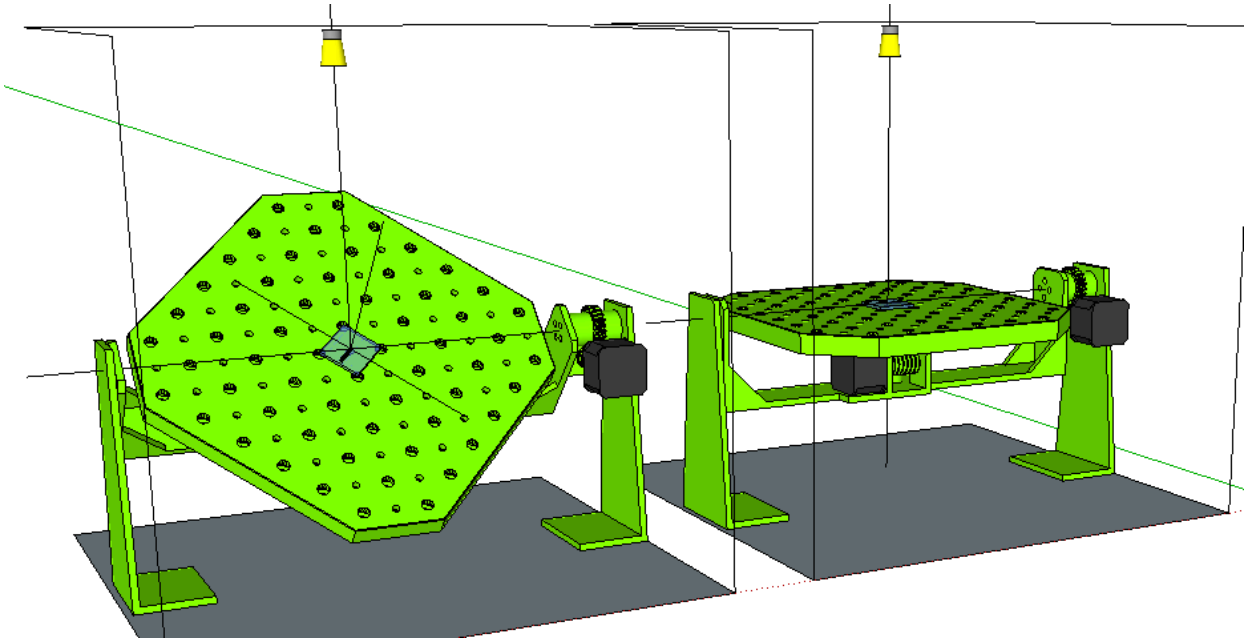
This design uses two motors to rotate the *Sample* around two axes: the horizontal and the perpendicular to the *Sample* surface.

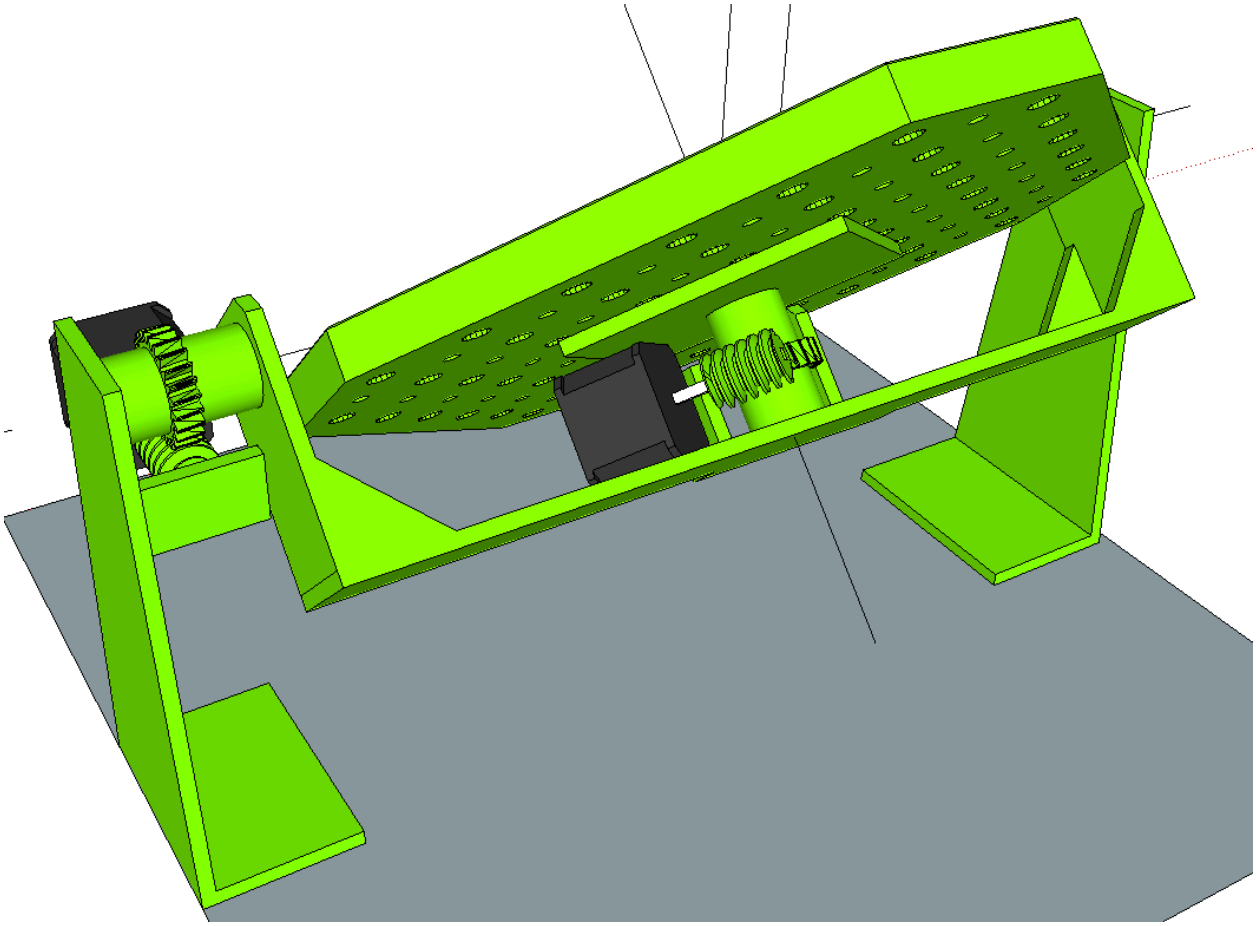
Link to 3D morel

The horizontal axis is fixed with reference to the lab, and performs a tilt rotation of 180º, 90º and -90º with respect to the horizontal position of the *Sample holder*.

The axis perpendicular to the *Sample* surface is not fixed with respect to the lab, it rotates 180º along the horizontal axis, using a *Motor* beneath the *Sample holder*.

Both motors transfer their rotational motion to the mechanical assembly through a worm gear (worm drive), made of a screw and a spur. Worm gears have very little backlash and provide high torque.





The advantage of this device is its simplicity for control. Moreover, it only uses two motors instead of three, which are required in the case of the *Spherical drive* device.

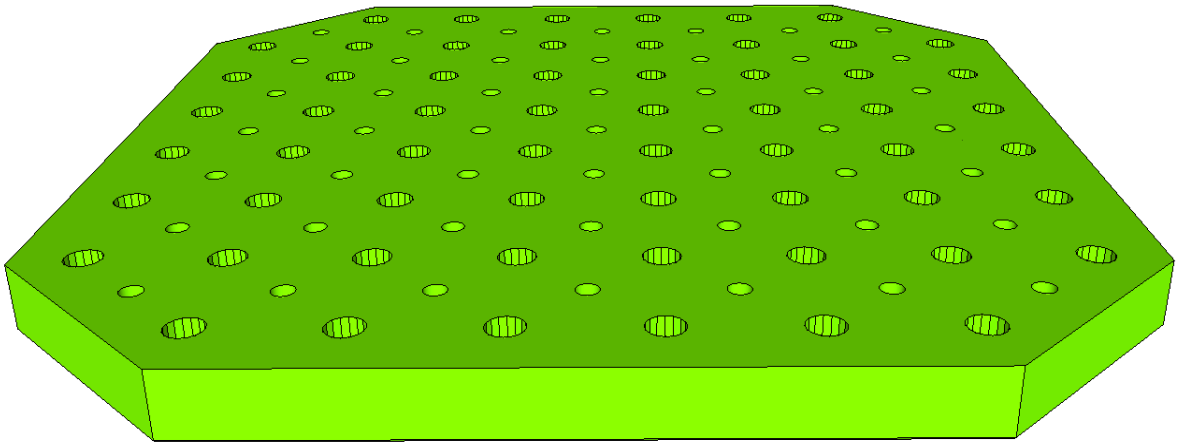
The disadvantage is that in order to avoid shadows from other devices placed on the *Sample holder* (for example the *Micromanipulators*) the user might have to manually rotate the *Sample*.

If this design is chosen, we will need to expand on the design elements in the prototyping stage.

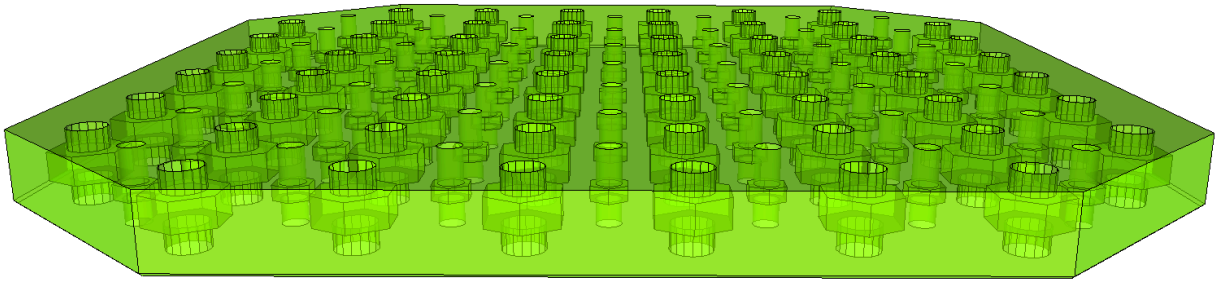
Sample Holder

Breadboard style

This *Sample holder* is a 3D printed thick plate, with an internal honeycomb structure to reduce its weight and to increase rigidity. The surface has a pattern of standard holes (same standard used by Thorlabs) to allow the user to attach other devices on it. Every hole contains a nut of a proper size. The nut is added to the structure during the 3D printed process. The *Sample holder* is printed in its horizontal position, the printing process is paused when it reaches a certain height to manually insert the nuts, and it is restarted after that to complete the process.



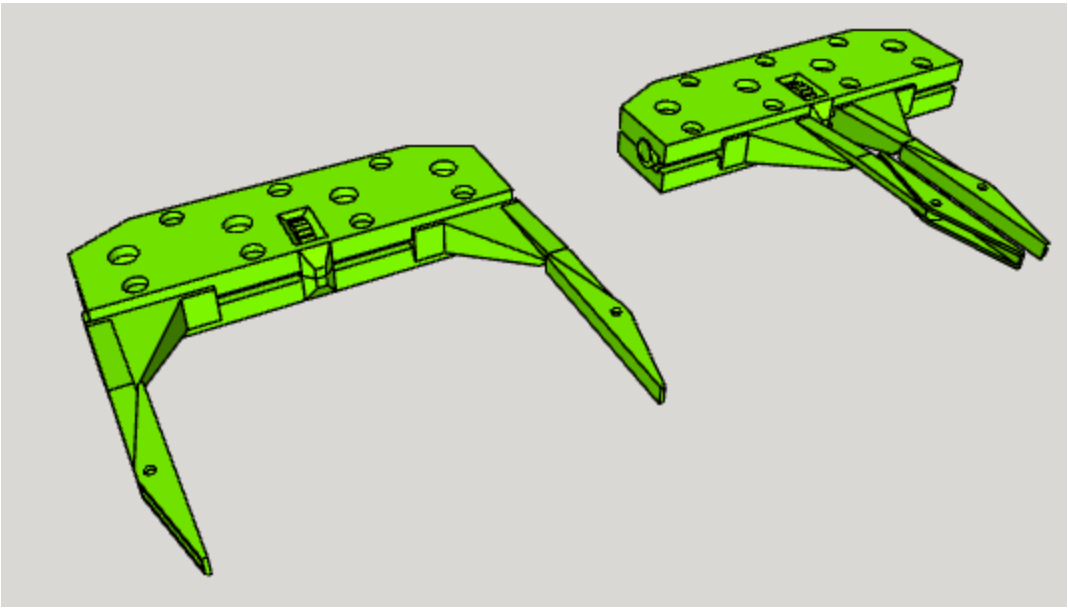
The picture below shows a transparency of the holes and the nuts in the *Sample holder*.



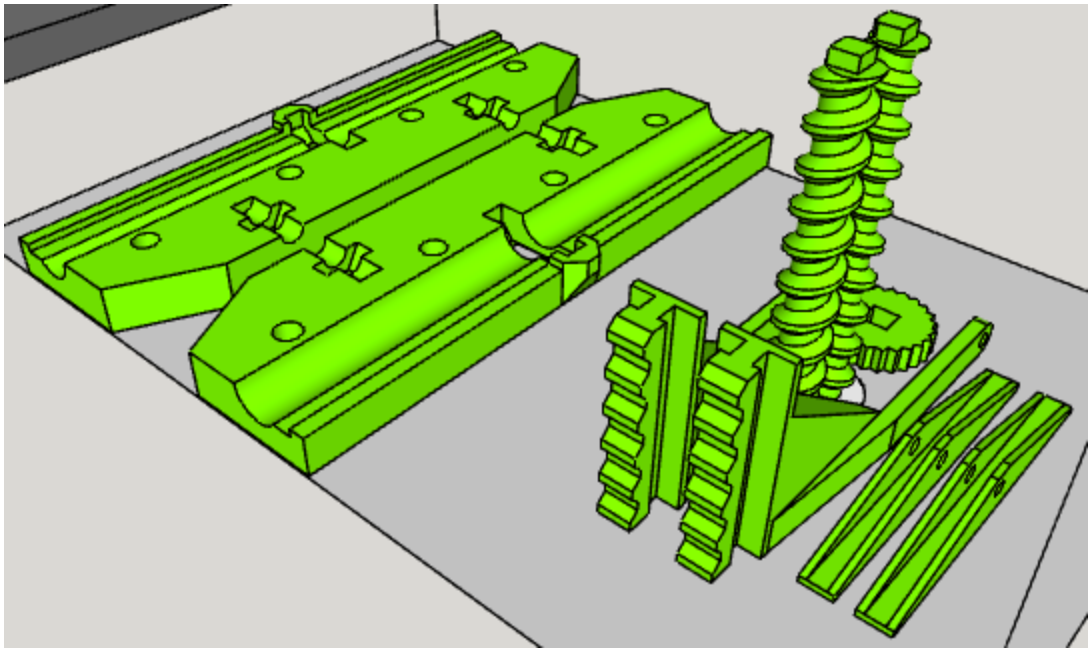
Clamp style

This Sample holder is an adaptable size clamp. The two sides that hold the sample can rotate, to adapt for different shapes of the Sample substrate.

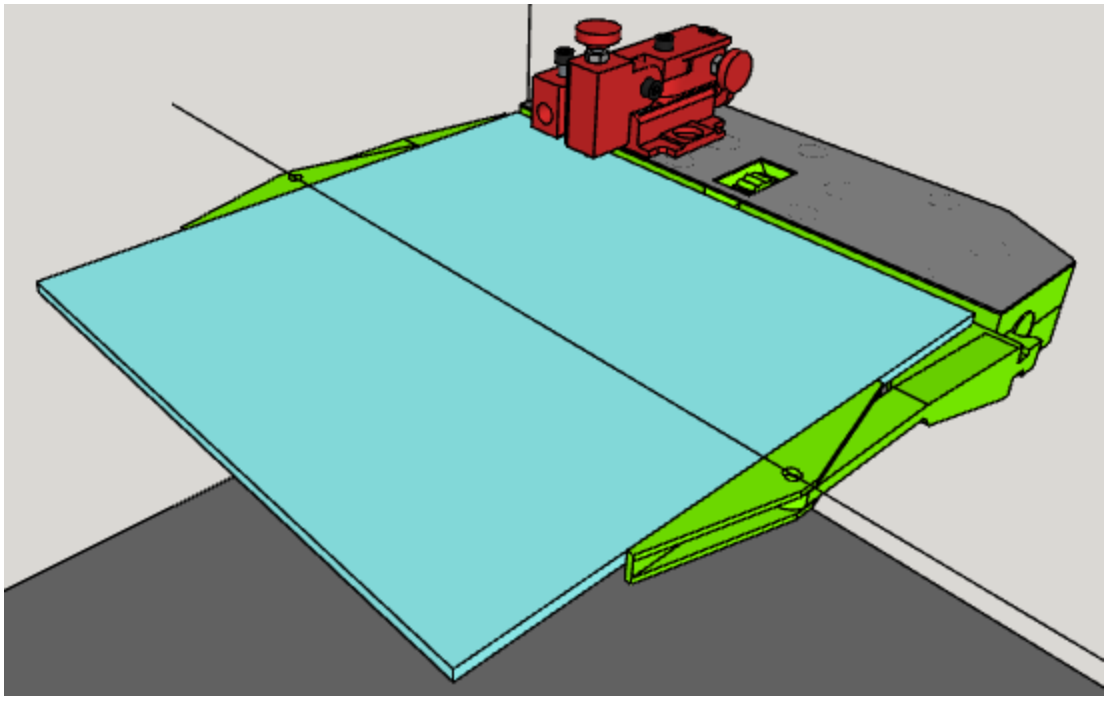
The assembled part, in the opened and closed position



Individual components

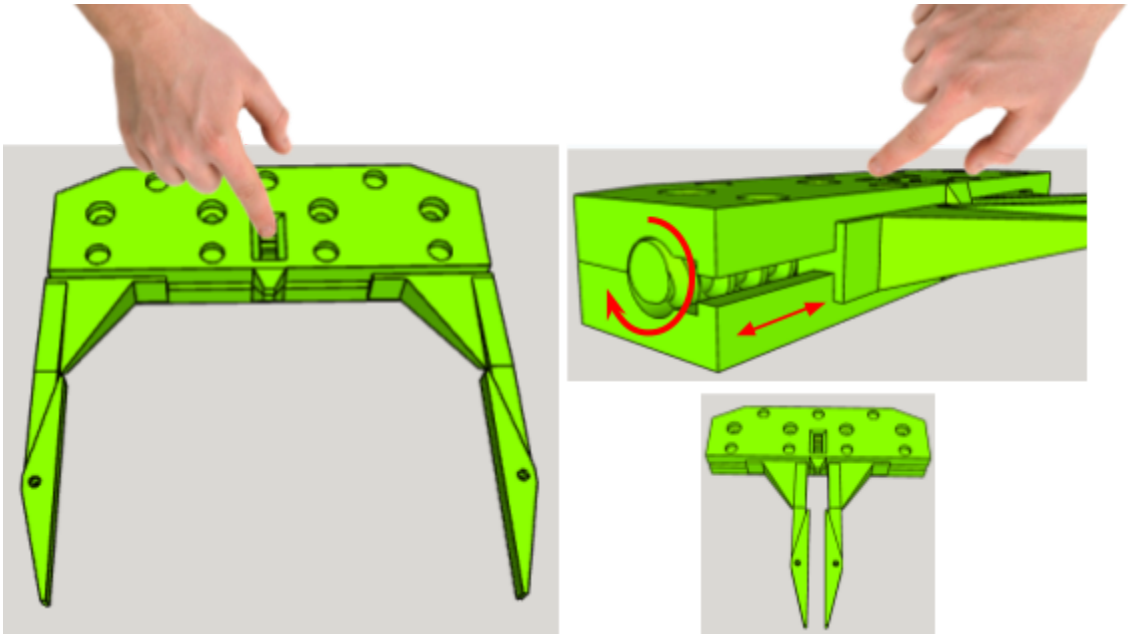


The clamp in operation



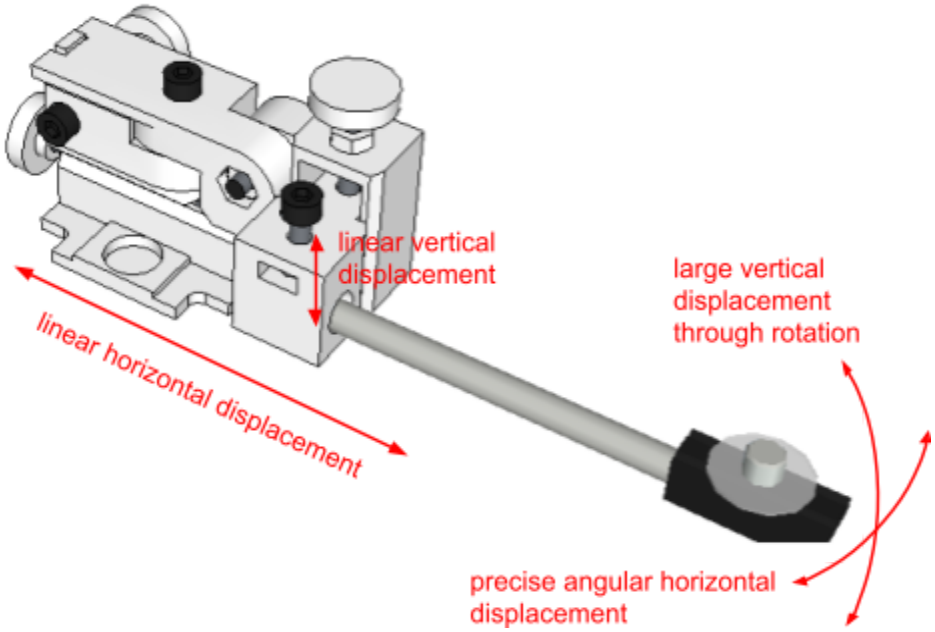
Operation of the clamp

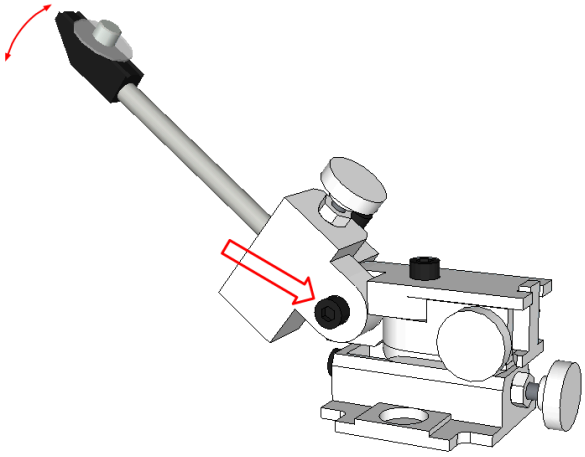
The clamp can be opened and closed using a knob, as in the picture below. The two sides of the clamp are activated by an internal screw. This entire device can be easily fixed on the gimbal with two screws.



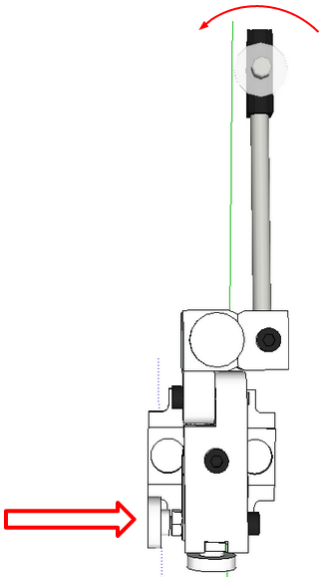
Micromanipulator

The micromanipulator is a three axes device used to fix the gold wire contacts on the sample.
Link to 3D model.

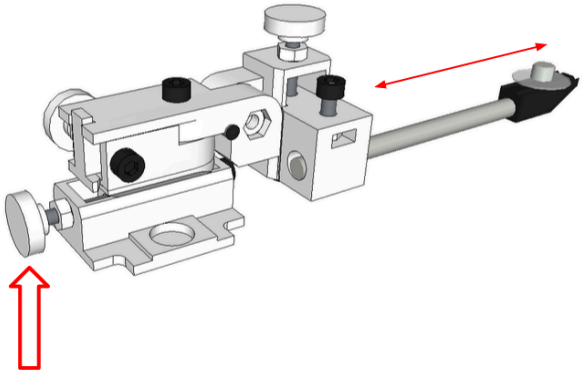
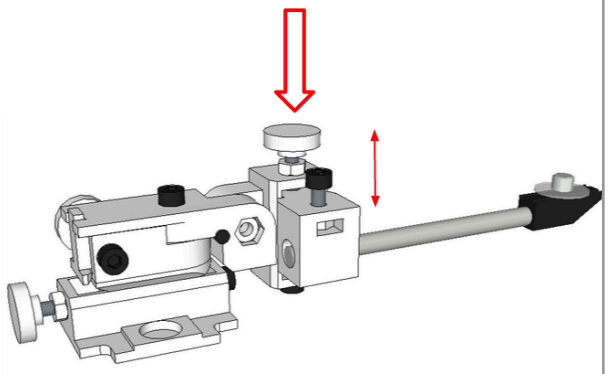




Large vertical displacement through rotation, actuated manually. Used to distance the gold wire from the *Sample* for adjustments.

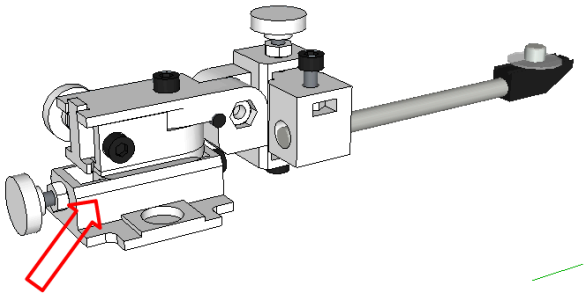
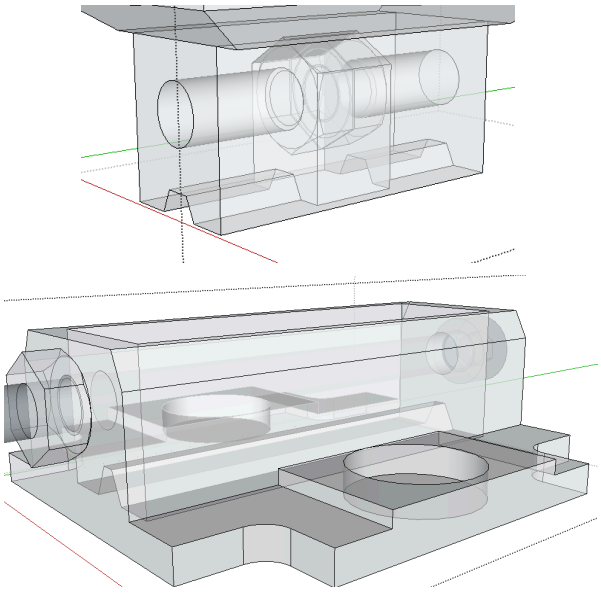


Precise angular horizontal displacement, actuated with an adjustment screw. Used for fine position adjustments of the gold wire on

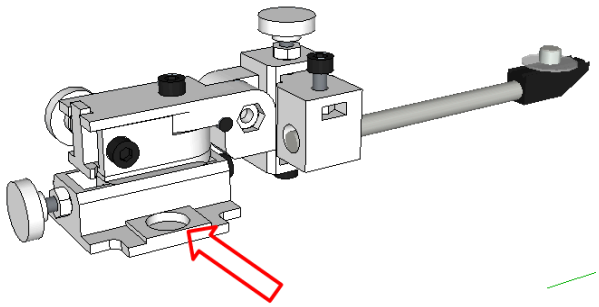
	the *Sample*'s connector.
 Linear horizontal displacement, actuated with an adjustment screw. Used for fine position adjustments of the gold wire on the *Sample*'s connector.	 Linear vertical displacement, actuated with an adjustment screw. Used for fine position adjustments of the gold wire on the *Sample*'s connector.

Some important features to consider

V-shaped rails for linear motion

The linear stages are sliding on a V-shaped groove. This design was chosen to reduce the requirement for high tolerance of the 3D printing process. 	

Magnetic fixtures

The Micromanipulators are fixed on the *Sample holder* with magnets, on a metallic plate. Round holes have been designed into the base of the Micromanipulator to accept small, cylindrical shaped, strong neodymium magnets.	

NOTE: requires some improvement and testing.

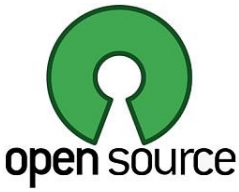
Clamps

Clamps are used to fix the *Sample* on the *Sample holder*, or various wires. The clamps can be fixed themselves on the *Sample holder* with screws, in the holes provided within the *Sample holder*.

The clamps need to be designed and 3D printed.

Software Design

Software Requirements

The software component of this project will be released under the GNU General Public License (GNU-GPLv3)	
The linux software components should abide to the advice and guidelines on designing effective interfaces with GTK+. More information on the Gnome Developer Center.	
Open Source Software practices should be respected as much as possible. We encourage remixers to document their work so that anyone else can collaborate or continue at any entry point in time. For anyone interested in further reading, here is a detailed guide: Producing Open Source Software.	

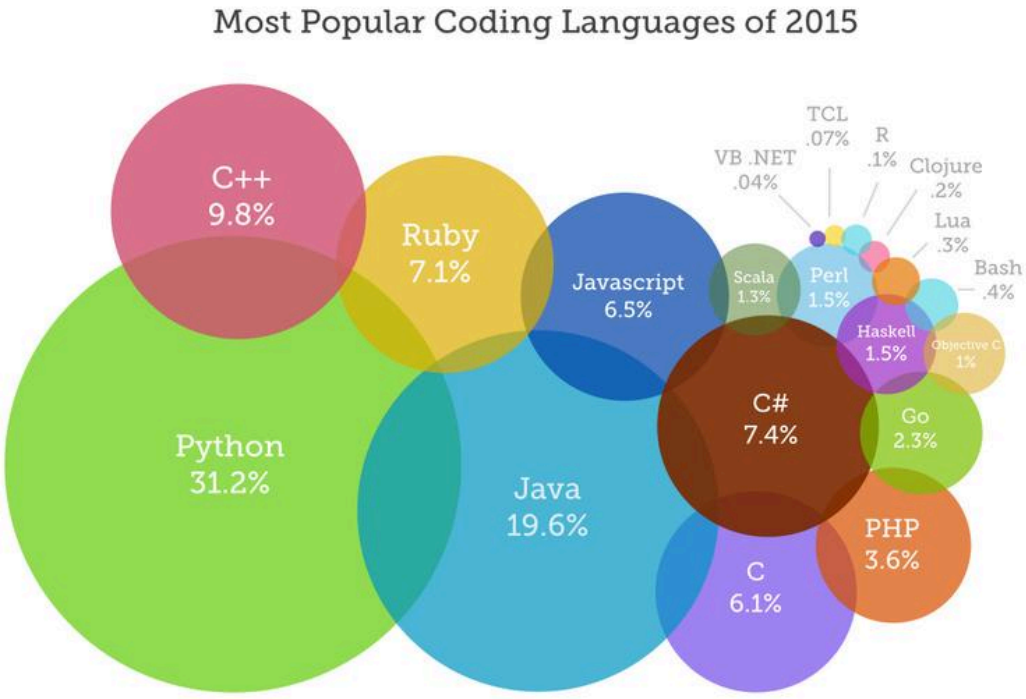
Developer Guidelines

Communication for the software component in the PV Characterization can be found in the software development project forum. Discussions, ideas, feedback and bug reports can be made public in the forum; It is also encouraged to announce any additions made to this document there.	
Deliverables for this and further milestones shall use the git distributed revision control system. The project is hosted on github.com under the Sensorica/PVCharacterization repository. The **Fork and Pull** method will be practiced for this project, as it is better suited for open source software projects and larger collaborative efforts.	

Linux Application

Programming Language: Python 3.4.3

Python is a widely used general-purpose, high-level programming language.
Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. Python can also be easy to pick up whether you're a first time programmer or you're experienced with other programming languages, and it is the most popular and socialized of all languages at the moment.



Python 2 or Python 3?

This common debate is coming to an end as Python 3 and most of its libraries have reached a greater level of maturity. We can finally say that projects taking life in 2015 can use Python 3 unless they're being deployed in an environment that you don't control. These environments may impose a specific version or require the use of a specific package or utility that doesn't yet have a released version compatible with Python 3. This is not the case for this software application.

GUI Development tools: GTK+ and GLADE

GTK+ or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off tools to complete application suites. **License:** GNU Lesser General Public License	
Gui Designer: Glade Interface Designer is a Rapid Application Development tool to enable quick & easy development of user interfaces for the GTK+ toolkit with additional components for the GNOME desktop environment. Glade is programming language – independent, and does not produce code for events, but rather an XML file that is then used with an appropriate binding. **License:** GNU General Public License	

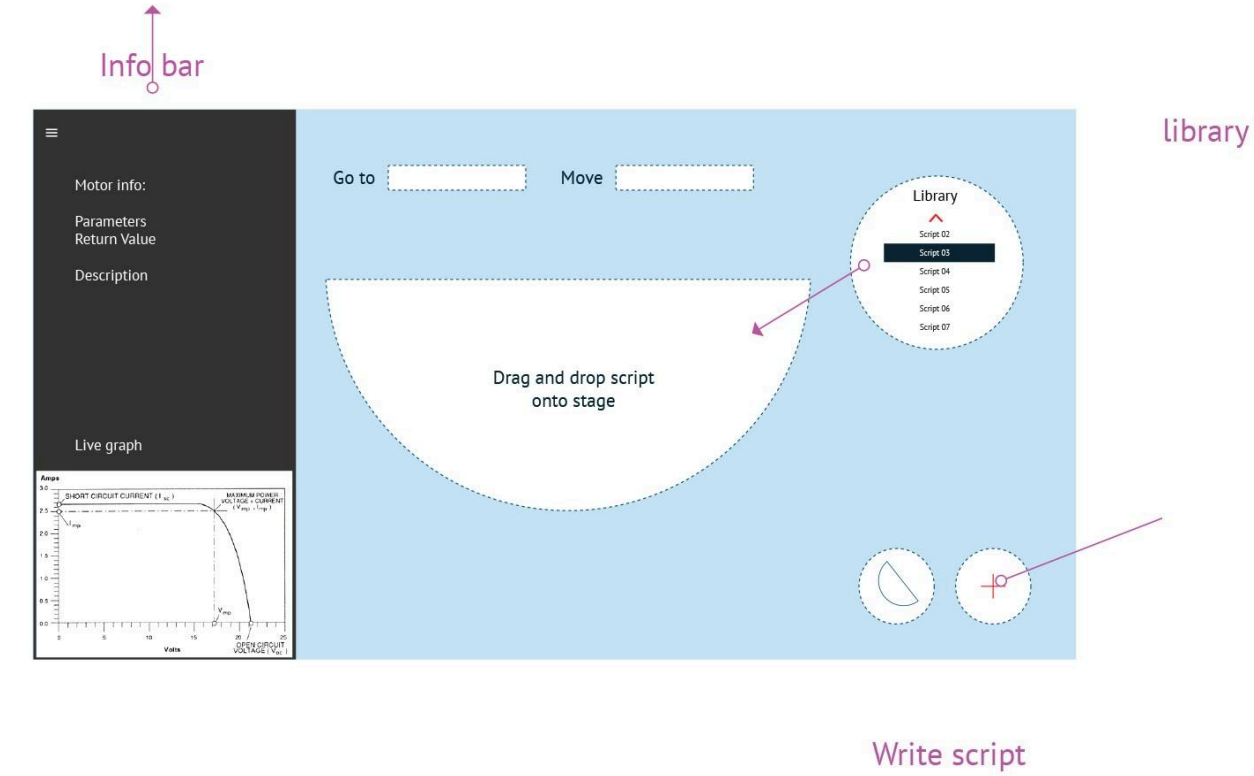
UI/UX Design

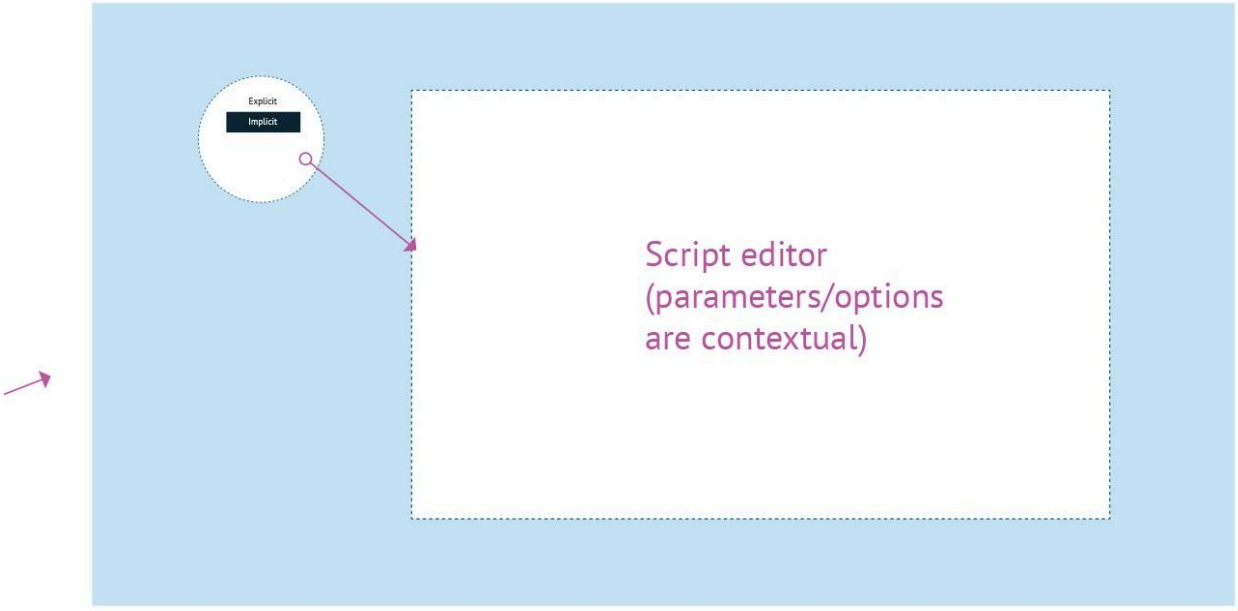
Here are the specific requirements to guide the design:

Commands	• Get position • Get motor information • Go home • Go to (enter tilt, rotation) • Go relative (degrees, speed, direction) • Go to recorded position • Record position • Edit script • Get script - Play - Pause • Save script • Export (save) script to text file
Commands that need extra input	**Initiate scan** Single scan • Enter Start Position (tilt, rotation) • End position (tilt, rotation) • Enter speed Double scan (same parameters as single scan except it goes back to start position)

	● Enter Start Position (tilt, rotation) ● End position (tilt, rotation) ● Enter speed Non-continuous scan ● Enter Start Position (tilt, rotation) ● Enter End Position (tilt, rotation) ● Enter speed ● Enter set step ● Enter pause (i.e. wait 1 second, etc.) **Write Script** Implicit script ● go to position (absolute) ● pause ● move (relative) ● repeat step(s) Explicit script — > send complete list of individual commands, analogous to 3D printer G code.

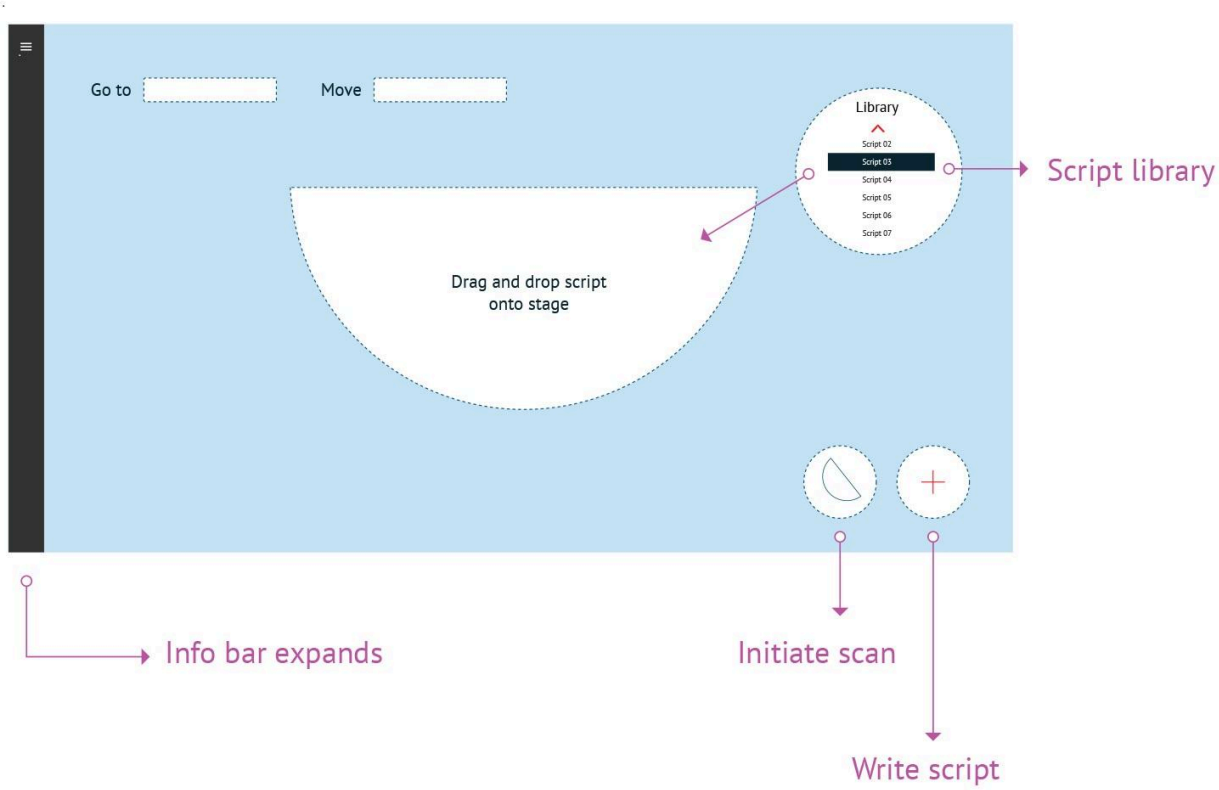
Here's a quick **concept** of the User eXperience. We want to demonstrate how the user can interact with the application in a way that is natural, makes use of contextual content as much as possible, and uses other non-traditional approaches to loading content such as drag and drop elements. There are three types of object areas on the main canvas: the **stage** (where the action takes place), the **library** (where scripts are stored), and **command type input fields**. You will notice the stage resembles the hardware; ideally we would use a rendering of it and if possible, once the script is dropped into it, it animates in real time. The rendering would have to be in the final version of the hardware. We have also included a first iteration of an expandable info bar as a useful place to keep content that the user might not want to see all the time.





We could approach the script editor in one of two ways: 1) Create a panel containing basic command types (i.e. move x steps, pause x seconds, repeat last x steps, go home, etc.) the user could drag and drop into the editor (followed by assigning a value to that

command through a dialogue box); 2) Take a more conventional approach, where



the user selects a series of commands through drop down menus from within the editor and then add the value through an input field.

The user could reorder and edit the values after they've been placed in the script editor.

When the script is complete, the user could run a simulation of the script, save the script and export it to a text file.

Arduino Firmware

Arduino Firmware Functionality

It is better to keep task definition on the PV software side and provide Arduino with the basic functions to perform these tasks. It is easier for developers to define new tasks in the PV software rather than in Arduino. Thus, Arduino functionality should be updated when there is a hardware change or an added PV software functionality that doesn't have low level function support.

Low level Functions

The following table shows the low level functions provided by the Arduino firmware. Some functions allow communication with external systems through a Serial over USB connection, while others are used for internal processing.

Category	Name of the function	Description	Priority
Stepper Motor Functions	Configure stepper motor	Allows changing the motor steps, from full step to microsteps	medium
	set direction	Parameters: motor id, direction Return Value: None Description: To change the direction of a running motor	high
	get direction	Parameters: motor id Return Value: direction Description: To retrieve the direction of a given motor	medium
	set speed	Parameters: motor id, speed Return Value: None Description: To change the speed of a running motor	high
	get speed	Parameters: motor id	medium

		Return Value: speed Description: To retrieve the speed of a given motor. This is achieved through the rate of change in encoder reading	
	set angle	Parameters: motor id, angle Return Value: None Description: To set the stepping angle of a running motor	medium
	get angle	Parameters: motor id Return Value: angle Description: To retrieve the stepping angle of a given motor in degrees. Assuming that there is a sensor to sense this data	low
	stop motor	Parameters: motor id Return Value: None Description: To stop a motor	high
	start motor	Parameters: motor id, direction, speed, angle Return Value: None Description: To start a motor	high
Feedback System Functions	Configure feedback system	Yet to develop: dependant on decisions made in prototyping	
	read sensor	Parameters: sensor_id Return Value: sensor_reading Description: To get a sensor reading from the feedback system	medium

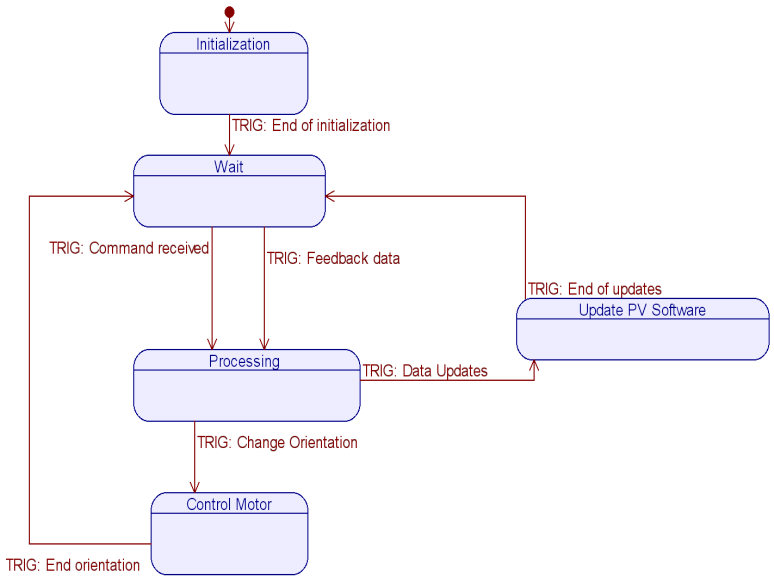
Data Structures

These are the essential data structures that should exist in the design. However, the proposed data type is for guidance and can be changed during implementation phase if there is a suitable reason. It is up to the programmer to use structs or simple variables.

Variables	Data type	Description
motor_id	byte	A unique identifier to identify each motor. Rotation motor =1 Tilt motor = 2
direction	byte	Determines the direction of motor movement
speed	integer	Identifies motor speed
angle	float	Identifies the motor stepping angle
status	byte	Determines the motor's status (i.e. ON=1, OFF=0)
sensor_id	byte	A unique identifier to identify sensors in the feedback system

State Diagram

The state diagram shows the available states of the arduino firmware. Moving from one state to another depends on the available triggers. These triggers can be hardware (i.e. interrupts) or software (i.e. change in flags or timers).



Created by QM (original file here)

Initialization state:

- Initializing the internal variables of the system (e.g. angle, speed, etc...).
- Performing startup checks by putting the motors in their initial state, and checking the sensor readings.

At the end of the initialization process, the “TRIG: End of initialization” is triggered to move to the “Wait” state.

Wait state: This is an idle state. During this state, If commands are received from the PV system (i.e. TRIG: Command received) or feedback information is received from sensors (i.e. TRIG: Feedback data), then the system switches to the “Processing” state

Processing state:

- Analyzing commands received from the PV software
- Analyzing data received from sensors

If the processing results show that data updates need to be sent to the PV software, then “TRIG: Data Updates” is initiated. If the data or commands received requires a change in motor status then “TRIG: Change orientation” is initiated.

Control Motor state:

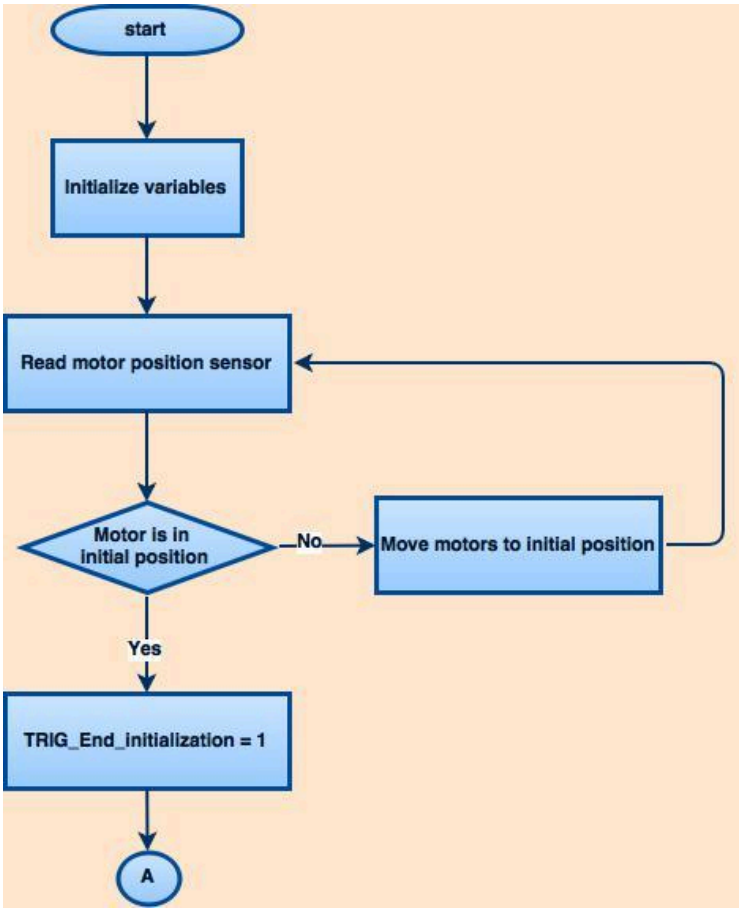
- Responsible for sending signals required to change motor status. “TRIG: End orientation” is initiated when changes are done.

Update PV software state:

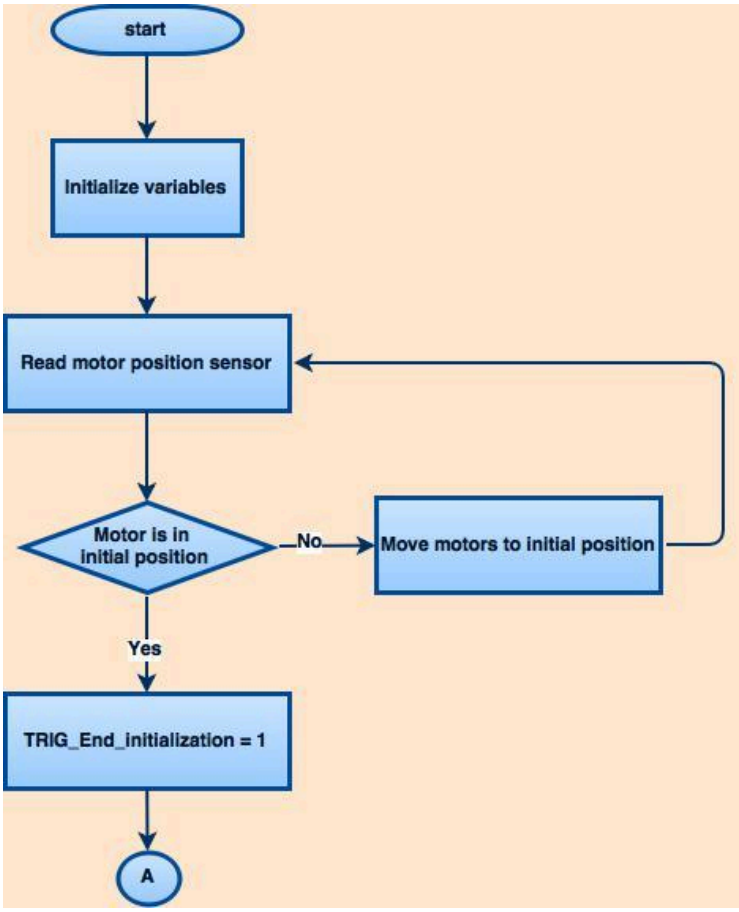
- Responsible for sending data to the PV software. When sending updates finishes, “TRIG:End of updates” is initiated and we go back to the wait state.

Flowchart

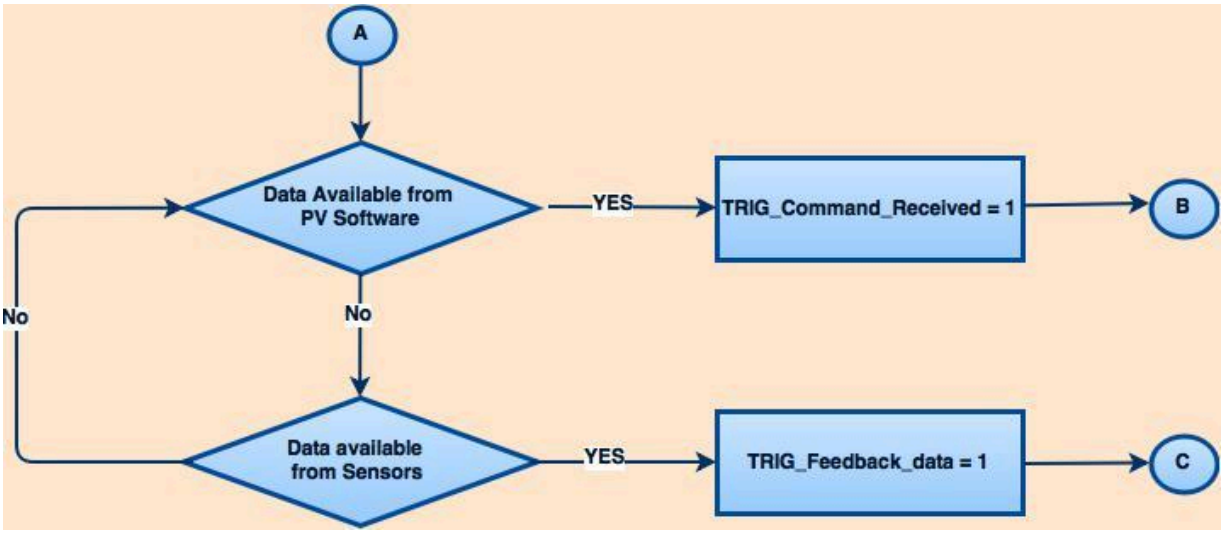
The functionality of each state in this diagram is further represented by a flowchart. The initialization state starts by initializing available variables such as default motor speed and initial angle. Moreover, it moves the motors back to their home position, and checks if the system is working properly.



Flowchart for the Initialization state

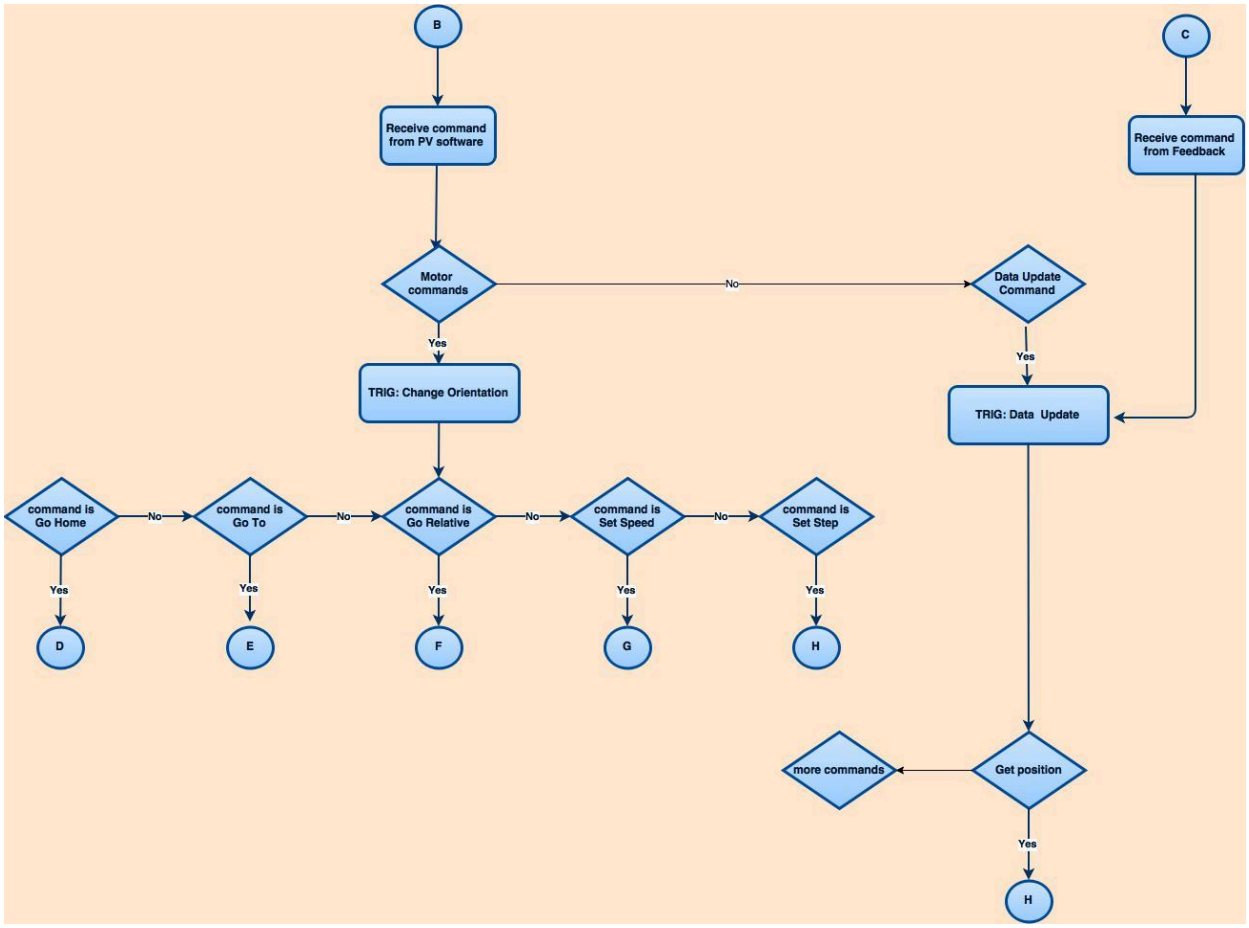


If commands are received from the PV software, then the suitable set of control functions is invoked. If new data is received from sensors, then the set of feedback functions is invoked.



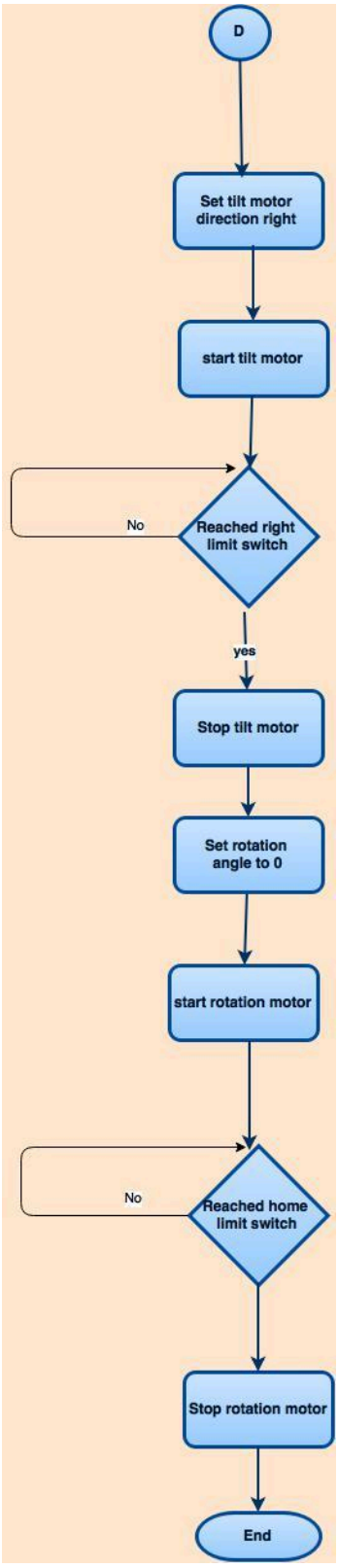
Flowchart for the Wait state

In the processing state, according to the received command,
certain low level functions are utilized.

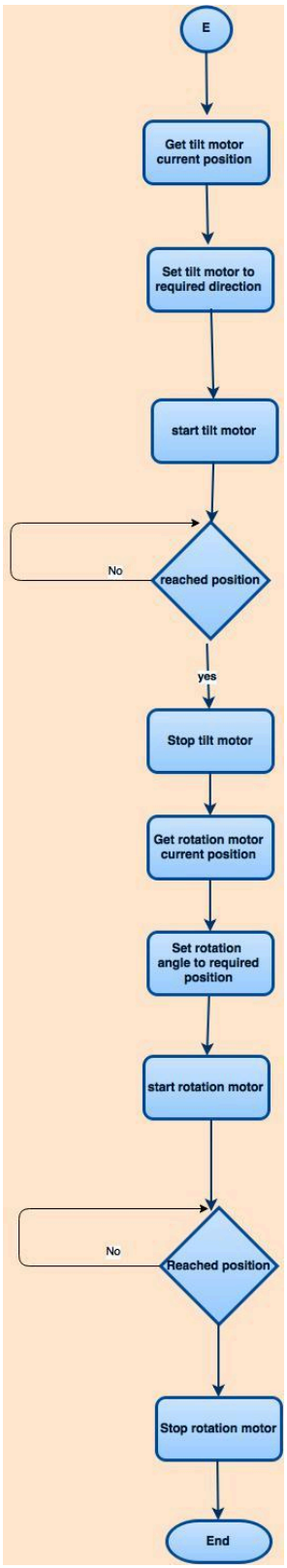


[Flowchart for the Process state]

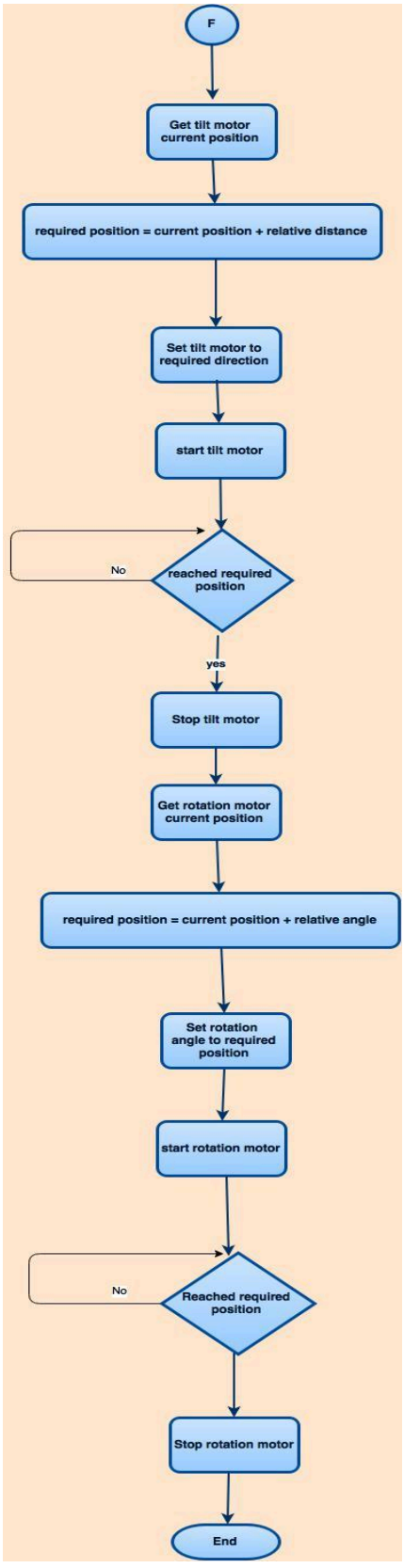
The **GoHome** function is used to move the Gimbal into its home position. This takes place multiple steps.The motors are moved until a reading from the right limit switch is received to indicate that the motors have reached their home positions.



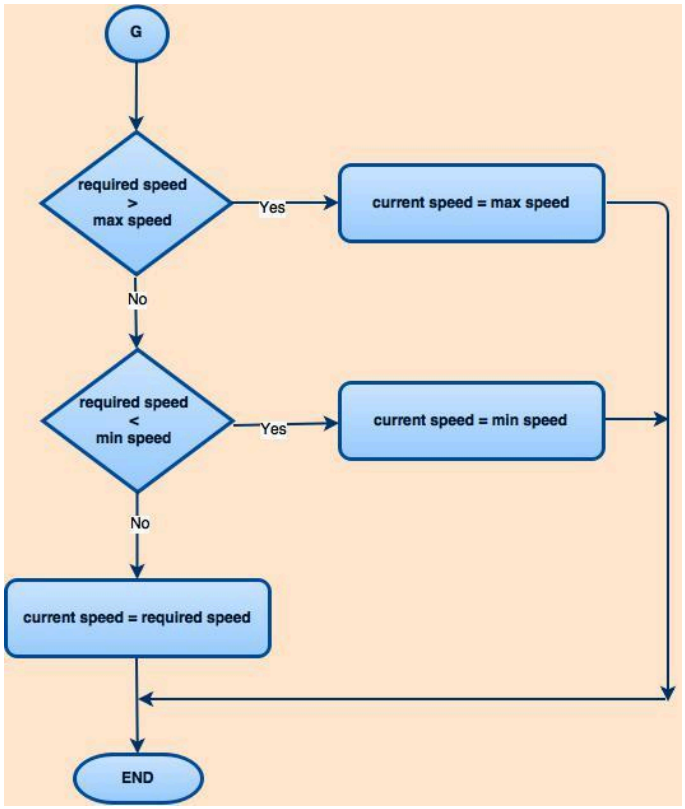
The **GoTo** function allows the motors to reach a specific position. It starts by getting the current position, then defines the moving direction, and finally moves the motor. The movement continues until the required position is reached. The same steps are performed for each motor.



The **GoRelative** function allows the motors to reach a position relative to the current position. It starts by getting the current position, calculates the required position, defines the required movement direction. The motor continues to move until the destination is reached. The same steps are performed for the rotation motor.



SetSpeed and **SetStep** Function



GetPosition Function