Project Report

Self-Levelling Surface

~ Adwait Dongre | Abhishek Shelar | Anil Dhaker | Raj Panchal

11/16/2012

Self-Levelling Surface

Members:

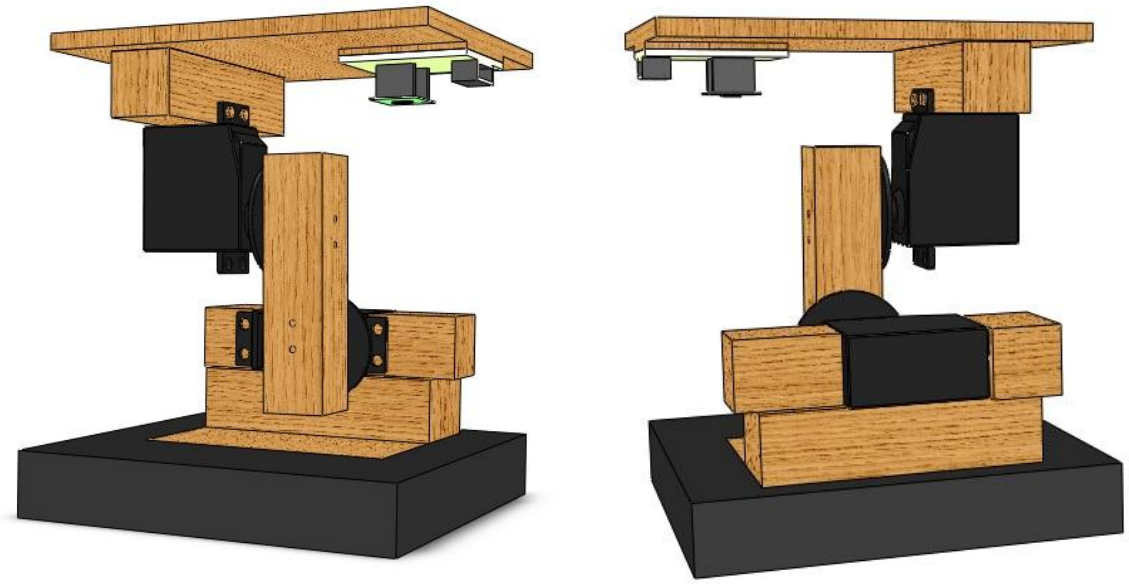
- Adwait Dongare (100260007)
- Abhishek Shelar (100260010)
- Anil Dhakar (100260014)
- Raj Panchal (100260003)

Introduction:

The objective of this project is to create a device capable of always maintaining a horizontal surface regardless of the orientation of the base. This is the implementation of a basic control system where the error is estimated by an input sensor (an accelerometer); a controller (the Arduino UNO) acts on this error to calculate the appropriate correction and forwards this correction to an actuator (a system of 2 servo motors).

As input for the system, the direction of ‘g’ would be taken as reference and the entire surface would be aligned so as to always keep ‘g’ along a pre-determined direction.

Hardware and Electronic Design:



CAD models made in SolidWorks

The system uses an accelerometer as the input sensor. The MMA7361L 3-axis accelerometer module from NEX robotics is used for this purpose. This device operates at 3.3V and would require appropriate voltage level corrections before being used with 5V devices like the Arduino UNO. This device is capable of operating in 2 modes: 1) 1.5g which features higher sensitivity and 2) 6g which features a larger range. Since our application does not involve large G forces and sensitivity is also critical, the device is used in the 1.5g mode. The output of the accelerometer is in terms of 3 analog voltages, each representing the ‘g’ component along a given axis.

The Arduino UNO using an Atmega328 is used as a controller. The Arduino operates at standard 5V voltages which can be supplied either directly via USB cable or using an external supply. The microcontroller has a built-in Analog-to-Digital Converter peripheral which is used to sense the analog outputs from the accelerometer. The device operates with a 16Mhz clock and an 8 bit processor which is sufficient for small computations like those required for this application. It has a number of PWM and digital IO pins that can be appropriately configured for output.

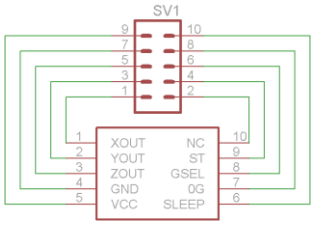
The actuators are a couple of servo motors, one a 3.4kg-cm Futaba motor and the another being a 6kg-cm Jetronics motor (both pin compatible). Both the motors operate between 4.5-6V, providing higher torques at higher voltages. Servo motors are controlled by a PWM signal. They require a positive pulse ranging from about 0.5ms to 1.5ms which is linearly mapped to the angle. Such a pulse must appear every 20ms in the control output.

The system has 2 independent surfaces, the base and the main surface. The surfaces are attached to each other by a mechanism of 2 servo motors such that the axes of rotation of the servos are perpendicular to each other. This ensures completely independent movement for the two surfaces in two axes. Assuming ‘g’ to be aligned along the Z axis, a combination of motions by the 2 servos can generate any rotation along the X and Y axes.

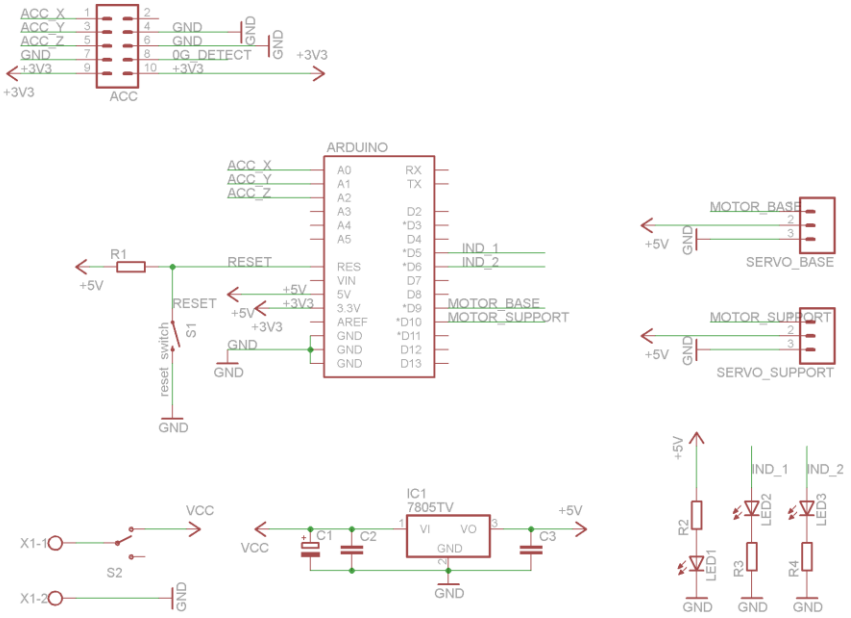
Since servo motors can move a maximum of 180 degrees, the alignment is made such that at the starting position (when the base and surface are almost parallel), both the motors are at the 90 degree position. This ensures corrections on both sides of the central point on both the axes. This also ensures that the system is always operating in the dynamic range of the servo corrections.

The accelerometer sensor is placed accurately at the bottom of the main surface such that in the ideal situation, ‘g’ would always be aligned along the most sensitive Z axis and one of the other axes is exactly aligned.

Custom circuit boards were printed for the accelerometer and the Arduino. A 10 pin FRC cable is used as a connection between the 2 modules. Servos are directly connected to the Arduino module.



The Accelerometer Module

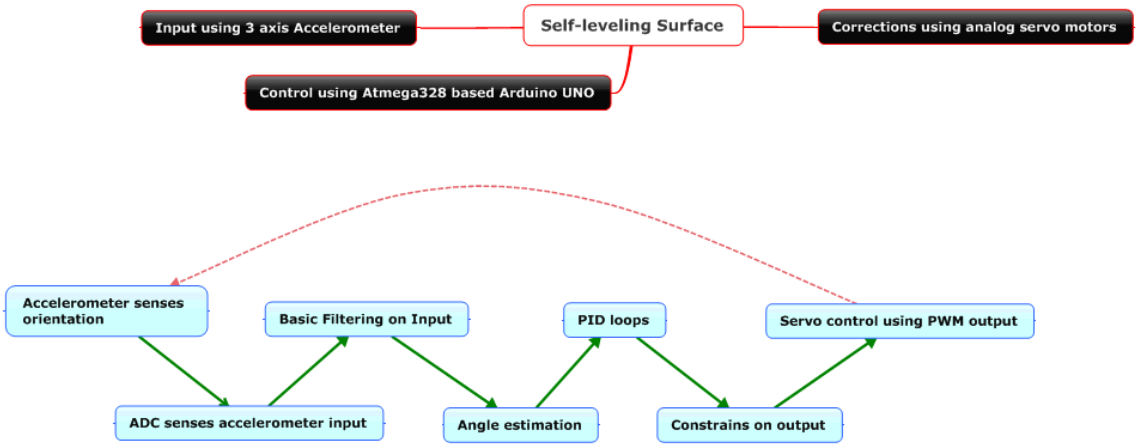


The Arduino module

Algorithm and Implementation:

Since the system uses Accelerometers, which are one of the most noisiest sensors, their outpts cannot possibly be used directly, especially for accurate applications. Thus a number of low pass digital filters are used on the input before it is forwarded to the control loop. This helps eliminate high frequency noise which can potentailly cause system unstability. An IIR version of low-pass filters is used:

$$\boldsymbol{X_o = \alpha X_{in} + (1 - \alpha) X_o}$$



Process flow of the Algorithm

The accelerometer output is an analog voltage between 0-3.3V. Depending on the datasheet specifications (and some practical experiments), a constant offset needs to be removed from these readings to determine the actual value of the 'g' component. This offset is determined by running a separate calibration code which just displays the current 'g' readings along all the axes. The system is placed in the equilibrium state (surface is exactly horizontal) and the readings are noted down to determine the offsets directly. To ensure the surface is exactly horizontal, another measurement device needs to be used (We used the calibrated and more accurate accelerometer in an Android phone). The conversion factor can be estimated from the ADC specifications and the Accelerometer specifications. The conversion can be given by this simple expression:

$$\boldsymbol{component\ of\ g = \frac{A_{in} - A_{offset}}{2^n} \times \frac{V_{ADC}}{sensitivity};\ A_{in}\ is\ the\ ADC\ reading}$$

The angle errors are given by simple arctan relations (due to the appropriate arrangement of sensors and the servo motors).

$$\boldsymbol{\delta\theta_i = \tan^{-1}\left(\frac{g_a}{g_z}\right)\ where\ \ a = y, x;\ i = 1, 2\ resp.}$$

These errors are then fed into a calibrated PID loop which gives the required corrections. Two PID loops run independently for the two servo controls. The implementation of I is done using a very low frequency low pass filter (the Fourier representations can be approximated to be the same as an integral). This avoids the problem of overshoot created when the system has entered a state of saturation and the integrated error just keeps on building up. We are likely to encounter this problem a number of times.

After the corrections are calculated, they are constrained within a particular range of angles to ensure that the surface never physically strikes any parts of the assembly on the base. This constrain angles is kept at a safe 30 degrees on each side of the central point in both the axes of rotation.

These constrained outputs are finally the corrections required to keep the system stable and are fed to the motors using PWM. The servo motors are directly controlled by using the Arduino Servo library which is optimized for such applications.

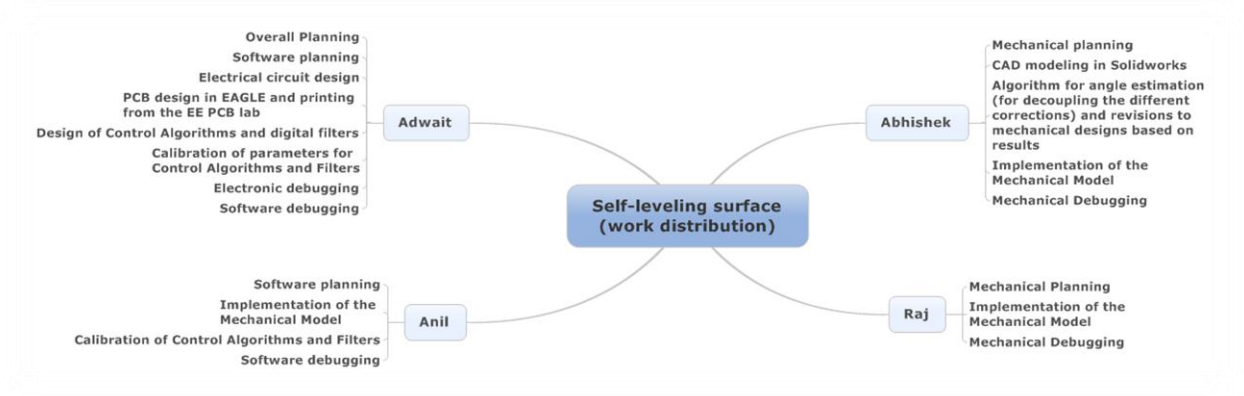
Calibration of the system is done by changing the parameters for the input low pass filters, the 3 PID constants and the offsets in the inputs. Offsets are calculated by experiment as already mentioned. The next parameters to fix are the PID constants using manual tuning methods. First K_p is chosen appropriately while keeping $K_d = K_i = 0$. Then depending on the amount of overshoot, K_d can be appropriately selected finally followed by K_i. The parameters of the input low-pass determine how sensitive the system is to fast changes (and thus also more sensitive to noise).

Planning and Work Distribution:

The entire project was split into 3 parts: 1) Electronic 2) Mechanical 3) Algorithms and work on each was done independently until the very end when each of the modules was combined. The Electronic and Mechanical designs were developed independently while keeping a track of the Algorithm requirements (a lot of which was done initially). When each of the sub-systems was assembled, they

were integrated in the very end and various calibrations were done to obtain optimal parameter values.

The work distribution among team members is given as follows:



Work Distribution

Known Issues:

There are some known issues with the present design which need to be fixed either by simple software changes as well as some major additions which require hardware level changes.

- The 2 servo motors together draw a lot of power and can cause a severe drop of the supply voltage, especially if it is drawn from a no-so-powerful USB port. This causes the Atmega to enter Brown-Out Protection mode (which basically shuts down the chip) but does not shut down the motors. This can cause some erratic behaviour especially if the system enters instability.
- PWM driven analog servo motors are used which have a max refresh rate of 50 Hz. However due to mechanical constraints, the actual refresh rate is much lower (~5 Hz). Thus no matter how well the software corrections may work, they cannot breach this limit.
- A lot of knocking may be seen, especially for large corrections in angles. This needs to be fixed by appropriate values of the PID parameters. However, optimal PID valued over a small but more accurate range would vary from the wide range but less sensitive regions.