Design of an Autonomous Self Correcting Platform Using Open Source Hardware

by

Manuel Juan Zeno

A Thesis Submitted to the Graduate Faculty of Rensselaer Polytechnic Institute in Partial Fulfillment of the Requirements for the degree of

MASTER OF MECHANICAL ENGINEERING

Approved:

__________________________________________
Ernesto Gutierrez-Miravete, Thesis Adviser

Rensselaer Polytechnic Institute
Hartford, CT

November, 2011
(For Graduation December, 2011)
## Contents

List of Figures ................................................................................................................ iv
Acknowledgement ........................................................................................................... v
Abstract .......................................................................................................................... vi
1. Introduction ................................................................................................................ 7
2. Methodology and Implementation ............................................................................. 14
   2.1 Electronic Hardware Assembly ....................................................................... 15
   2.2 The Arduino Platform ..................................................................................... 16
   2.3 Mechanical Control System Design ............................................................... 18
   2.4 Arduino Sketch Programming .......................................................................... 25
3. Testing and Results .................................................................................................... 28
4. Conclusion .................................................................................................................. 29
Appendix A ...................................................................................................................... 30
Appendix B ...................................................................................................................... 34
Appendix C – Analog Devices ADXL322 ................................................................. 36
Appendix D – Arduino Uno Datasheet ..................................................................... 50
Appendix E – Hitec 422 Servo ..................................................................................... 56
References ....................................................................................................................... 57
List of Figures

Figure 1 – Various platform types ........................................................................... 7
Figure 2 - St. Mary's U Platform .................................................................................. 9
Figure 3 - Devry New Brunswick .............................................................................. 10
Figure 4 - Self-Leveling Surface with Arduino .......................................................... 10
Figure 5 – U Berkeley Self-Leveling Platform ............................................................. 11
Figure 6 - Self-Leveling Platform ................................................................................ 11
Figure 7 - ECE572 Self-Leveling Platform ................................................................. 12
Figure 8 - Stewart Platform U of Adger, Norway ....................................................... 12
Figure 9 - Self-Leveling Platform for Autonomous Lawn Mower .............................. 13
Figure 10 - Platform Schematic .................................................................................. 14
Figure 11 - Electronics Schematic ............................................................................. 15
Figure 12 – Arduino Prototyping Board (Arduino Uno) ............................................. 16
Figure 13 – Arduino Integrated Development Environment ....................................... 17
Figure 14 - Servo Arrangement .................................................................................... 18
Figure 15 - Servo Body Brackets [19] ...................................................................... 19
Figure 16 - Servo "C" Brackets [19] .......................................................................... 19
Figure 17 - Digital Count Equivalent Acceleration .................................................... 21
Figure 18 - Accelerometer Gravity Measurement ....................................................... 23
Figure 19 - Angle of tilt vs. Acceleration .................................................................. 24
Figure 20 - Angle of Tilt Sensitivity Function ............................................................. 25
Figure 21 - Sketch Flow Chart ................................................................................... 26
Acknowledgement

I would like to extend a special thanks to the Arduino community. Their open and free resources allowed a person with awfully limited programming skills to create an exciting mechatronic wonder, if I do say so myself. I would also like to thank the developers of the Fritzing schematic software for their easy to use and free software. I would also like to thank Angela and my family for giving me the motivation.
Abstract

This paper demonstrates the feasibility of designing and building a self-correcting platform using inexpensive hardware and software (total design under $100US). The platform was designed using inexpensive materials, wood and aluminum sheet metal, and was controlled by an open source microcontroller, an accelerometer, and two servos. An Arduino microcontroller, hobby grade servos, and a two-degree of freedom (axis) accelerometer were used to create the controlled platform. The intent of the platform design is to maintain the platform at an initially selected angle while the support structure orientation changes. The software was written with logic to convert the digital data from the accelerometer to an acceleration magnitude vector. The magnitude was then compared to a predetermined mathematical function to infer the angle of tilt of the platform. The angle of tilt is then converted to angle of rotation for the servos to act on. Testing showed the platform to perform as expected. Although some error on the final angle was expected, the magnitude of the error observed indicated the platform design has a high sensitivity to low tolerance mechanical joints (slop). Overall the platform design was validated based on the positional accuracy of the platform given the low quality components used to create it. In other words, the platform performed greater than the sum of its parts.
1. Introduction

There are many benefits to an electro-mechanically controlled platform. This thesis proposes to design and build the controlled platform as a surface whose angle in relation to a user specified plane is controlled autonomously by electronic means while maintaining a budget of less than $100 US for all the hardware and software. The concept of the controlled platform is not unique. There are numerous commercial mechanical and electronically controlled platforms available today. Some examples are security camera pan/tilt/roll mechanisms, gimbaled nautical compasses, and scissor lifts (Figure 1). There are also patented platform designs such as the anti-motion sickness chair (US PAT 7,490,572B2) and the self-correcting stabilized platform (US PAT 6611662).

![Figure 1 – Various platform types](image)

The controlled platform is a popular university level project, where many teams from various colleges and institutions as well as independent parties (tinkerers) complete designs for credit. A web search of the terms “controlled platform” or “self-leveling platform” yields a number of different platform concepts of varying complexity and sophistication. Table 1 below lists the qualities of each design and the differences between them and the concept proposed by this paper. The author selected eight completed designs for comparison. These designs were selected using their similarity in form and function to the proposed platform as the main criteria. Note that some of the platforms do not have a complete design description or parts list. Those will be compared using the pictorial or video evidence available via the references.
<table>
<thead>
<tr>
<th>Platform Description</th>
<th>Budget</th>
<th>Electronics</th>
<th>Hardware Sophistication</th>
<th>Build Team</th>
<th>Angle Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-level</td>
<td>Mid-level</td>
<td>Arduino Digital Servos</td>
<td>Mid-level Two axis</td>
<td>St. Mary’s U. single person team</td>
<td>none</td>
</tr>
<tr>
<td>Automotive self-leveling</td>
<td>Mid-level</td>
<td>PIC18F452 micro-controller</td>
<td>Mid-level ~Three axis</td>
<td>Devry New Brunswick ~3 person team</td>
<td>none</td>
</tr>
<tr>
<td>Self-leveling Surface</td>
<td>Low-level</td>
<td>Arduino</td>
<td>Low-level Single axis</td>
<td>Independent party</td>
<td>none</td>
</tr>
<tr>
<td>Self-leveling platform</td>
<td>High-level</td>
<td>ezDSP</td>
<td>High-level Three axis</td>
<td>Berkeley 4 person team</td>
<td>none</td>
</tr>
<tr>
<td>Automotive self-leveling</td>
<td>Mid-level</td>
<td>unknown</td>
<td>Mid-level Two axis</td>
<td>Unknown Institution</td>
<td>none</td>
</tr>
<tr>
<td>Self-leveling platform</td>
<td>Mid-level</td>
<td>dsPIC</td>
<td>Mid-level Single axis</td>
<td>Unknown Institution</td>
<td>none</td>
</tr>
<tr>
<td>Stewart Platform</td>
<td>High-level</td>
<td>unknown PC Control</td>
<td>High-level Six axis</td>
<td>U of Adger, Norway</td>
<td>likely</td>
</tr>
<tr>
<td>Self-leveling platform</td>
<td>High-level</td>
<td>PC Control via Hitachi SH2</td>
<td>High-level Two axis</td>
<td>Unknown, Germany</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 1 - List of Platform Projects

The criterion for budget is approximate based on the available data. Low-level is approximately $100 USD or below, mid-level is <$500, and high-level is >$500. The electronics column names the microcontroller package used as well as the servo type, if available. The hardware sophistication is meant to describe the build quality and cost of the design. This column is purely subjective, as no cost data was available for the others. However, the build quality and cost can be estimated by studying the pictures or videos. Low-level is wooden, with analog servos. Mid-level, is some wood and metal
parts, with digital servos, and High-level is highly designed (likely machined) parts. Note that none of the platforms comparable to the proposed allow the user to control the level angle of the platform. The section below informally references the source of the platform design as well as some comments by the author summarizing the differences between the proposed concept and the referenced platform. Platform 9 is the author-designed concept.

Figure 2 - St. Mary's U Platform

Figure 2 is a self-leveling platform similar in function and hardware to the author’s. The control system is Arduino based and the platform mechanics are linkage based. The major difference between the referenced platform and the proposed is the ability to control the angle of the platform manually. This platform has a higher hardware budget with digital servos. [1]
Figure 3 illustrates a platform design, which is completely different in form and function. There is no explanation or link to a paper. Based on the video, the platform is a self-leveling automotive suspension system. [2]

Figure 4 - Self-Leveling Surface with Arduino

The platform pictured in Figure 4 is a single axis self-leveling platform. There is no manual control of the platform angle. The platform suffers from accelerometer feedback (repetitive swinging). This is due to the accelerometer placement on the platform without some attenuation algorithms in the source code. [3]
The platform in Figure 5 is sophisticated three axis self-leveling platform. A team of four students created the platform with a large budget and access to manufacturing resources. The platform is designed with two square metal frames and one metal platform connected with and swivel on rods. The team used an ezDSP board from Texas Instruments ($320). [4]

Figure 6 is a three-axis platform. Unfortunately, there is no documentation about the design of the platform. From the video, it was found to lack independent angle control and appears to suffer from servo lag and, due to little or no averaging of the accelerometer data, servo stutter. [5]
Figure 7 - ECE572 Self-Leveling Platform

Pictured in Figure 7 is a single degree of freedom platform controlled using a dsPIC board. There is no further documentation on this design. The platform is controlled via a single servo directly driving the rotation in one axis. Given the nature of the project and the name of the file, this is most likely a final project in an undergrad or grad level electronics class. [6]

Figure 8 - Stewart Platform U of Adger, Norway

This Stewart platform is a large budget six degree of freedom controlled platform. The platform sitting atop the six jacks are controlled via hydraulics in a synergistic manner. This type of platform is outside the scope of this thesis and is referenced for information only. These platforms are commonly used in flight simulators and professional astronomy. [7]
This platform (Figure 9) is designed to stabilize the navigational system of an autonomous lawn mower. The platform is similar in function to the one pictured in Figure 2. The design uses a sophisticated control system to maintain the sensor array level. The sensor array allows the lawn mower to navigate predetermined boundaries and needs to be maintained at a constant orientation for accuracy.

The differences between the platform design suggested here and the referenced projects are the cost and the ability of the user to input a specific platform angle. Teams of students with larger amounts of funding and infrastructure (machine shops, etc.) developed the majority of the examples seen at other institutions. This allows for greater hardware quality and mechanism sophistication. Many of the referenced projects use machined parts and expensive electronics packages to build the control systems.

This thesis will demonstrate the concept using inexpensive open source commercial microcontroller to control a wooden platform in two degrees of freedom (axis), pitch and roll.[9] The platform will remain level relative to a selected position using inputs from an accelerometer with servo motors used to control the mechanics of the platform. The “zero” position will be selected by the user using two potentiometers, each adjusting one axis. The platform was built with a $100.00 budget.
2. Methodology and Implementation

The self-correcting platform consists of two platforms (Figure 10), the “top” platform (smaller wood piece), which is autonomously controlled, based on initial user input, and a “bottom” platform (larger wood piece), which is where the accelerometer is installed and the angle of tilt is measured.

![Platform Schematic](image)

**Figure 10 - Platform Schematic**

An Arduino Uno board, ADXL322 accelerometer, two HITEC 422 servos, and a 9V battery control the upper platform tilt angle. The materials selected for this design were purposely inexpensive. The one exception is the ball bearing brackets (Figures 15,16). The brackets were purchased with ball bearings to ensure the assembly would not bind while moving. The following sections describe the mechanics, electronics, and programming aspects of the platform.
2.1 Electronic Hardware Assembly

The Arduino board connections to the accelerometer and servos are illustrated in Figure 11. Normally, the Arduino board and other accessories can be powered from the universal serial bus (USB) connection used to program the board.

![Figure 11 - Electronics Schematic](image)

However, the servos used for this board require more power than the Arduino board can supply from the 5V power pin. Because of this, the servos are powered using an external power source, a 9V battery. According to the HS-422 servo datasheet the servos are rated to 6V max, so the 9 volts from the battery are converted to 5 volts via a 5V, 1A power regulator. The regulator has enough power capacity to drive both servos at once with low load.

The potentiometers used to manually control the platform are supplied with 5V from the Arduino board. It is the reduction in voltage the potentiometers supply to the analog pins which signals the board to send an angle signal to the servos. The accelerometer is powered by the Arduino board’s on-board 3.3V power. The ADXL322 accelerometer will send different signals through the analog pins depending on the supply voltage.
2.2 The Arduino Platform

The Arduino platform is an open source electronic prototyping system. It is composed of two parts, the Arduino Uno board and the Arduino IDE (Integrated Development Environment). The Uno board (Figure 12) is designed to provide an easy to use human changeable pin interface to the Atmel® AVR® ATmega microcontroller [10], the heart of the Arduino hardware. The AVR® microcontrollers from Atmel® use the C language for programming and are commercially available and very popular with hobbyists and electronics enthusiasts. Arduino builds on this by adding simplicity to the hardware interface and an easy to use software package.

![Figure 12 – Arduino Prototyping Board (Arduino Uno)](image)

Arduino is meant to be used as a physical computing platform [10]. That is, to use the electronic hardware to interface with humans using sensors and actuators controlled by software executed by a computer. The groundwork of this paper focuses on the physical computing aspect of Arduino. The internal functionality and architecture of the ATmega328 microcontroller and the Arduino board is outside the scope of this thesis and as such will not be discussed further.

The Arduino IDE (Figure 13) is the software environment used to create the programs, called “sketches,” that will be executed by the Arduino hardware. The IDE uses a modified C language compiler to build, translate, and transmit the code to the microcontroller board. The AVR® microcontrollers are typically programmed using the C language. Since C can be somewhat difficult to learn for the typical hobbyist, the IDE
makes a combination of modified C and Arduino specific commands available to the user. The IDE provides the simplicity for less proficient users but supports advanced users who are knowledgeable in C. The IDE also has a set of additional code functions called libraries. Libraries extend the commands available to provide capabilities not available in the core Arduino language [11]. In essence, a library allows the user to perform seemingly complex functions using a small set of commands. For example, this paper will focus on controlling servo motors. The SERVO library is called within the sketch to simplify the code. All libraries used are provided in the appendix along with the complete servo sketch source code.

Figure 13 – Arduino Integrated Development Environment
2.3 Mechanical Control System Design

The servos supporting and controlling the top platform are arranged to control the pitch and roll. Note that pitch and roll are only subjective directions and are used to describe the motion. Since the platform has no “front” or “back” the terms pitch and roll are meaningless descriptions of the rotations about the X and Y axis. A close up view of the servo arrangement and axis’ is illustrated in Figure 14.

![Servo Arrangement](image)

Figure 14 - Servo Arrangement

This servo arrangement is a more efficient use of space and limits the amount of slop compared to linkages and hinges as seen in other designs. The Y axis servo is controlled by the Y axis data stream from the accelerometer and vice versa for the X axis. Note that the X axis servo motion is in-plane with the photo while the Y axis would rotate out of plane.

The servos are installed into aluminum brackets illustrated in Figure 15. These brackets allow the servo to be installed onto the bottom and top platforms. The servo “C” brackets (Figure 16) are then used to attach the servo bodies, via the body brackets, to themselves and the platforms. The “C” brackets attach to the body and to the rotating
servo horn. The combination of these two brackets allow for the platforms to rotate independently in two axis.

A dual axis accelerometer was the chosen sensor to measure the tilt of the lower platform in two axes. The data is read from two different channels and processed in the
same manner regardless of direction. The two analog data streams provide the Arduino logic with the necessary information to maintain the top board level to the chosen plane. The ADXL322 accelerometer was chosen because it is a relatively inexpensive sensor with a reasonable amount of sensitivity (~420 mV/g). The sketch program, using three functions developed specifically for this application, determines the orientation of the board. These functions translate the digital data read by the Arduino board into acceleration in terms of Earth’s gravity (Gs), Angle of Tilt, and correction for sensitivity drop-off.

The Arduino board measures the voltage output from the accelerometer in an analog fashion. The signal read from the analog pins is converted to a digital signal using a 10-bit analog to digital converter. As such, the analog pins read 0V to 5V in integer values between 0 and 1023 units.

The following procedure was developed to convert the analog signal read by the Arduino unit into acceleration:

The sensitivity of the accelerometer is 420mV/g nominal with an input voltage of 3.3V [12]. The Arduino unit reads the accelerometer analog signal input voltage with a 10 bit analog to digital converter and a 5V analog reference (from the Arduino unit), giving a resolution of:

$$\frac{5V}{1024} = 0.00488V = 4.88mV \text{ per count}$$

The sensitivity of the accelerometer, 420mV/g, indicates that 1G of acceleration is equal to 420mV at the accelerometer output pin. When the accelerometer is tilted, the measurement of acceleration due to gravity can be positive or negative, depending on the direction assigned as positive. Note that the positive or negative direction of tilt is irrelevant for this application. It is the absolute value of tilt angle that matters. In view of the fact that the accelerometer and lower platform can be tilted in any direction, the entire range of -1G to +1G is required. It is necessary to find the zero G voltage output from the accelerometer so as to identify the digital count equivalent. That is
accomplished by calculating the ratio of the accelerometer sensitivity and the resolution of the analog pin, then subtracting it from the sensitivity.

\[
\frac{420 \text{ mV/g}}{4.88 \text{ mV}} = 86 \text{ counts per G}
\]

\[
420 - 86 = 334
\]

The middle point of the sensing range between -1G and +1G is 334 counts. Therefore, the digital count value of 420, 334, and 248 at the pin equates to -1G, 0G, and +1G respectively. From this information a linear function was derived to allow the microcontroller program to convert any analog value from the accelerometer and equate it to a ratio of acceleration due to gravity. Figure 17 illustrates the linear function derived.

![Acceleration per Digital Counts](image)

Figure 17 - Digital Count Equivalent Acceleration

The equation of the line, where the y axis is acceleration in Gs, allows the sketch to

\[
y = -0.0116x + 3.8832
\]
translate the digital output from the analog to digital converter into Gs.
The Arduino program measures the tilt of the base board using the magnitude of gravity measured by the accelerometer as an input vector to determine the orientation of the base board in space in two rotational directions. This is possible because as the accelerometer is tilted, the component of acceleration due to gravity in the sensing direction varies. Figure 18 illustrates how the accelerometer indirectly measures tilt. At position A the accelerometer’s sensitivity axis is parallel with the gravity vector. The accelerometer measures $\alpha_N = G$, or the acceleration due to gravity. At position B the sensitive axis is tilted to angle $\theta$. The acceleration measured at the now tilted sensitive axis is

$$\alpha_{N\theta} = G \times \cos \theta$$

The normal force measured by the accelerometer is reduced by the cosine of the angle of tilt. Since the accelerometer is sensitive in only one direction, as the accelerometer is tilted, the component of the acceleration due to gravity in the sensing axis changes. This rationality allows the program to infer the angle of tilt based on the magnitude of the acceleration due to gravity measured.
For example, if the accelerometer’s sensitive axis is perpendicular to the surface of the earth, the acceleration measured would be

\[ \alpha_{N\theta} = G \times \cos 90^\circ = 0. \]

This means that the accelerometer’s sensitive axis no longer measures any acceleration due to gravity. Conversely, if the sensitive axis is parallel to the gravity vector, then

\[ \alpha_{N\theta} = G \times \cos 0^\circ = G \]

And the accelerometer measures 1 G of acceleration.

The function of tilt angle to the acceleration measured by the accelerometer is not linear for large angles. This is due to the sensitivity drop-off as the gravity measurement values become smaller. The behavior of the function becomes asymptotic at large angles as seen in Figure 19. The best approximation to the behavior of the accelerometer is the inverse sine (arcsine) of the ratio of the measured acceleration to the acceleration due to gravity [12].
The “Angle of Tilt” function relates the ratio of the acceleration due to gravity with the measured acceleration from the accelerometer to the angle of tilt in degrees.

\[
\text{Angle of Tilt (Deg)} = \sin^{-1} \left( \frac{\text{Measured Acceleration (Gs)}}{1 \text{ G}} \right) \times \frac{180}{\pi}
\]

The non-linear behavior exhibited by the accelerometer is due to the sensitivity drop-off. The ADXL322 accelerometer is most sensitive to tilt when the plane of the circuit board is perpendicular to the gravity vector. When the accelerometer’s X,Y plane is oriented perpendicular to the gravity vector (near its +1g or −1g reading), the change in output acceleration per degree of tilt is negligible. When the accelerometer’s X,Y plane is parallel to the gravity vector, its output is approximately 17.5 mg per degree of tilt. At 45°, its output changes to about 12.2 mg per degree of tilt [12]. So as the degree of tilt increases the resolution of the measurement declines. The degree of sensitivity drop-off is taken into account when using the angle of tilt function. A linear function (Figure 20) was produced using the two sensitivity values provided the ADXL322 datasheet (17.5 mg/degree for <45° and 12.2 mg/degree for >45° to illustrate the reduction in sensitivity.
The platform control sketch was written in the Arduino code. The control sketch was written to accommodate the accelerometer inputs, servo control, and mechanical limits of the platform as well as the accelerometer sensitivity and servo lag. The sketch logic is a simple loop. The design of the platform has no feedback loop and as such has no fine control (angle error correction) of the final platform position. A feedback loop would require a second accelerometer to confirm the upper platform is at the intended orientation. The top platform angle controlled by the sketch has some error due to linkage slop, servo dead zone, and accelerometer error. Linkage slop is defined as the play within the swivel joints. Linkage slop is not a concern for this concept because higher tolerance ball bearing joints were selected. The servo slop however, remains the highest error inducing factor in the design. Due to budget constraints, inexpensive analog servos were procured. The Hitec 422 is an analog servo with a plastic gear transmission and brass bushing shaft support. Although the analog servos are inherently slower and less precise than digital servos, they are adequate for the demonstration purposes of this concept. Knowing the limitations of the hardware, the sketch software
was written such that the behavior of the servos and the platform control is equivalent to a mechanical linkage. For example, a mechanical linkage may be used to control a door opening mechanism. The linkage will always maintain the mechanical position of the door, but will be subject to the slop (error) due to pivots and gears. Although the servos introduce error, the platform will always keep the correct orientation. This is because the servo angle is wholly a function of the accelerometer tilt data. Note the flow chart illustrated in (Figure 21).

Controlled Platform Sketch Flow Chart

![Figure 21 - Sketch Flow Chart](image-url)
The sketch begins with the administrative tasks of calling the servo library, initializing variables, and assigning the Arduino board input and output pins. These are initialized only once within the sketch and are not part of the main program loop. The main program loop contains all the commands for reading the accelerometer data and converting the data to useful information for the servos. Appendix A contains the main platform sketch as well as ancillary sketches used to debug the platform sketch. Appendix B contains the sketch used to quantify mechanical limits of the platform. Those mechanical limits were measured and the angles were then used in the final version of the platform sketch. Without these mechanical limits, the platform mechanics can bind and cause current spikes, which may damage the electronic hardware.
3. Testing and Results

The testing is meant to determine the accuracy of the leveling capability of the platform when aligned parallel to the surface of the earth. The actual transit time is not a success criterion. The platform functionality was tested using a bubble level. The testing process began by aligning the upper platform parallel to the earth (or perpendicular to the earth's acceleration due to gravity vector) using a bubble level. The lower platform was then rotated in one and then both axis while observing the bubble level. The testing was completed with a low mass object, the bubble level, and a higher mass object (two ‘D’ size batteries and the bubble level).

In both test cases the platform performed as designed. A quantitative measurement was completed and the bubble level results showed the platform’s accuracy to be approximately 90%. This accuracy value was determined by observing how much of the bubble remained within the “level” limits marked on the bubble tube. When a heavier object was placed on the level, the accuracy improved to about 95%. Some extreme angles (within the mechanical limits) were tested and the platform maintained consistent behavior. It was also observed that the platform was more accurate when correcting in one direction versus another on the same rotation axis. This phenomenon is likely due to the flexibility of the wooden components and the mechanical slop of the joints the servos act on. The results of the testing proved that the platform design functioned as designed and intended. Videos of the platform tests are available in Reference 9.
4. Conclusion

This paper demonstrated the feasibility of the platform design using inexpensive hardware and software; total design under $100US. The platform was designed using inexpensive materials, wood and aluminum sheet metal, controlled by an open source Arduino microcontroller, an accelerometer, and two servos. Software was written with logic to convert the digital data from an accelerometer to an acceleration magnitude vector. The magnitude was then compared to a predetermined function to infer the angle of tilt of the platform. The angle of tilt is then converted to angle of rotation for the servos to act on. Testing showed the platform to perform as expected. Although some error on the final angle was expected, the magnitude of the error observed indicated the platform design has a high sensitivity to low tolerance mechanical joints (slop). Overall the platform design was validated based on the positional accuracy of the platform given the low quality components used to create it.

A few unintended behaviors were discovered during the testing phase that limits the functionality of the platform. The platform exhibits a phenomenon the author calls “motion tilt”. This phenomenon occurs when the platform is exposed to acceleration other than gravity. This behavior can be remedied by implementing other accelerometers to detect such movements and a software update with logic used to ignore “in plane” motion.

The platform was also found to have some lag between the readings and the tilt correction. This behavior is not easily corrected. It is due a number of variables like the latency of the servos and the accelerometer reading averaging function. However, the success criterion for the platform design was the actual correction of the tilt angle, not the speed at which it was corrected. There are many performance improvement opportunities that can be implemented with simple modifications to hardware and software. Further development opportunities include the control of other degrees of freedom, such as yaw, cancellation of all non-gravitational acceleration to reduce the effects of motion tilt, and use of digital servos with metal gears will improve the latency and reaction time of the platform.
Appendix A

Servo_correction.pde

// Self leveling platform sketch

#include <Servo.h> //Servo library
Servo yservo; // create servo objects to control the servos
Servo xservo;

int accelpinx = 0; // Defines the Accelerometer Pin X & Y dir
int accelpiny = 1;
int xpotpin = 2; // Correction potentiometer pins
int ypotpin = 3;
int xval = 0; // Variable used to represent the accel pin value
int yval = 0;
int xsval; //Start servo angle at 90 DEG.
int xaval; //accel angle variable
float xaalal;
int xtilt;
int ytilt;
int ysva; //Start servo angle at 90 DEG.
int yaval; //accel angle variable
float yaal;
const int numReadings =10;
int xreadings[numReadings]; // the readings from the analog input
int xindex = 0; // the index of the current reading
int xtotal = 0; // the running total
int yreadings[numReadings]; // the readings from the analog input
int yindex = 0; // the index of the current reading
int ytotal = 0; // the running total
int xpotcor = 0; // level correction using potentiometer
int ypotcor = 0; // level correction using potentiometer
void setup()
{
// Serial.begin(9600); // For debug purposes - printing the val and sval
    yservo.attach(9); // Assign servo on pin 9 to the yservo object
    xservo.attach(10);
    //averaging function
for (int thisReading = 0; thisReading < numReadings; thisReading++)
{
    xreadings[thisReading] = 0;
    yreadings[thisReading] = 0;
}
}

void loop()
{
xval = analogRead(accelpinx); // read the accel value 0 - 1023
yval = analogRead(accelpiny);
xpotcor = analogRead(xpotpin);
ypotcor = analogRead(ypotpin);
xval = constrain(xval,240,428); //limits glitches and outliers due to noise
yval = constrain(yval,240,428); //limits glitches and outliers due to noise
xaaval = (xval*-0.0116) + 3.883;
yaaval = (yval*-0.0116) + 3.883;
xtilt = (asin(xaaval/1)*180)/3.1416; //determine angle of tilt in degrees
ytilt = (asin(yaaval/1)*180)/3.1416; //determine angle of tilt in degrees
xtilt = constrain(xtilt,-36,56); //constraint due to mechanical limits
ytilt = constrain(ytilt,-56,30); //constraint due to mechanical limits
xpotcor = map(xpotcor, 0, 1023,-20,40); //
ypotcor = map(ypotcor, 0, 1023,-20,40); //

// ******************** START Averaging accelerometer analog data
xtotal= xtotal - xreadings[xindex];       // read from the sensor:
xreadings[xindex] = xtilt;     // add the reading to the total:
xtotal= xtotal + xreadings[xindex];       // advance to the next position in the array:
xindex = xindex + 1;
if (xindex >= numReadings)          // if we're at the end of the array...
xindex = 0;     // ...wrap around to the beginning:
xaval = xtotal / numReadings;     // calculate the average:
// ******************** FINISH Averaging accelerometer analog data

// ******************** START Averaging accelerometer analog data
ytotal= ytotal - yreadings[yindex];    // read from the sensor
yreadings[yindex] = ytilt;       // add the reading to the total
ytotal= ytotal + yreadings[yindex];    // advance to the next position in the array
yindex = yindex + 1;
if (yindex >= numReadings)     // if we're at the end of the array...
yindex = 0;     // ...wrap around to the beginning:
yaval = ytotal / numReadings;    // calculate the average:
// ******************** FINISH Averaging accelerometer analog data

xsval = 90 + (xaval) + xpotcor;
ysval = 90 - (yaval) + ypotcor;
//add the tilt angle to the default vertical position
xservo.write(xsval); // Send signal to x servo
yservo.write(ysval); // Send signal to y servo
delay(15);  // delay to allow servos to move (ms)
// delay can be improved using digital servos
}

/* debug section - print accel parameters to determine if correct
Serial.print("raw x");
Serial.print("\t");
Serial.print("raw y");
Serial.print("\t");
Serial.print("X Gs");
Serial.print("\t");
Serial.print("Y Gs");
Serial.print("\t");
Serial.print("xtilt");
Serial.print("\t");
Serial.print("ytilt");
Serial.print("\t");
Serial.println();
Serial.print(xval);
Serial.print("\t");
Serial.print(yval);
Serial.print("\t");
Serial.print(xaaval);
Serial.print("\t");
Serial.print(yaaval);
Serial.print("\t");
Serial.print(xtilt);
Serial.print("\t");
Serial.print(ytilt);
Serial.print("\t");
Serial.println();
*/
}
Appendix B

platform_limit_test.pde
//Used to determine mechanical limits of platform assembly to hard
code them in the platform leveling sketch.

#include <Servo.h>

Servo servx; // create servo object (from library)
Servo servy; // create servo object (from library)
int potx = 0; // analog pin 0 used to connect the potentiometer
int poty = 1;
int xval;    // variable to read the value from the analog pin
int yval;    // variable to read the value from the analog pin

void setup()
{
    Serial.begin(9600);
    // For debug purposes - printing the val and sval to serial USB
    servx.attach(9);
    // attaches the servo on pin 9 to the servo object
    servy.attach(10);
    // attaches the servo on pin 10 to the servo object
}

void loop()
{
    xval = analogRead(potx);
    // reads the value of the potentiometer (value between 0 and 1023)
    xval = map(xval, 240, 430, 0, 179);
    // scale it to use it with the servo (value between 0 and 180)
    yval = analogRead(poty);
    // reads the value of the potentiometer (value between 0 and 1023)
    yval = map(yval, 240, 430, 179, 0);
    // scale it to use it with the servo (value between 0 and 180)
servx.write(xval);
// sets the servo position according to the scaled value
// servy.write(yval);
// sets the servo position according to the scaled value
Serial.println(xval);
Serial.print("\t");
Serial.println(yval);
delay(15);
// waits for the servo to get there
Appendix C – Analog Devices ADXL322

**FEATURES**

- Small and thin: 4 mm × 4 mm × 1.45 mm LFCSP package
- 2 mg resolution at 60 Hz
- Wide supply voltage range: 2.4 V to 6 V
- Low power: 340 μA at V_S = 2.4 V (typ)
- Good zero g bias stability
- Good sensitivity accuracy
- X-axis and Y-axis aligned to within 0.1° (typ)
- BW adjustment with a single capacitor
- Single-supply operation
- 10,000 g shock survival
- Pb Free: Compatible with Sn/Pb and Pb-free solder processes

**APPLICATIONS**

- Cost-sensitive motion- and tilt-sensing applications
- Smart hand-held devices
- Mobile phones
- Sports and health-related devices
- PC security and PC peripherals

**GENERAL DESCRIPTION**

The ADXL322 is a small, thin, low power, complete, dual-axis accelerometer with signal conditioned voltage outputs, which are all on a single monolithic IC. The product measures acceleration with a full-scale range of ±2 g (typical). It can also measure both dynamic acceleration (vibration) and static acceleration (gravity).

The ADXL322’s typical noise floor is 220 μg/√Hz, which allows signals below 2 mg to be resolved in tilt-sensing applications using narrow bandwidths (<60 Hz).

The user selects the bandwidth of the accelerometer using capacitors Cx and Cy at the XOUT and YOUT pins. Bandwidths of 0.5 Hz to 2.5 kHz can be selected to suit the application.

The ADXL322 is available in a 4 mm × 4 mm × 1.45 mm, 16-lead, plastic LFCSP.

---

**FUNCTIONAL BLOCK DIAGRAM**

![Functional Block Diagram](image)

Figure 1.
# ADXL322

## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifications</td>
<td>3</td>
</tr>
<tr>
<td>Absolute Maximum Ratings</td>
<td>4</td>
</tr>
<tr>
<td>ESD Caution</td>
<td>4</td>
</tr>
<tr>
<td>Pin Configuration and Function Descriptions</td>
<td>5</td>
</tr>
<tr>
<td>Typical Performance Characteristics (V_S = 3.0 V)</td>
<td>7</td>
</tr>
<tr>
<td>Theory of Operation</td>
<td>11</td>
</tr>
<tr>
<td>Performance</td>
<td>11</td>
</tr>
<tr>
<td>Applications</td>
<td>12</td>
</tr>
<tr>
<td>Power Supply Decoupling</td>
<td>12</td>
</tr>
<tr>
<td>Setting the Bandwidth Using C_x and C_y</td>
<td>12</td>
</tr>
<tr>
<td>Self-Test</td>
<td>12</td>
</tr>
<tr>
<td>Design Trade-Offs for Selecting Filter Characteristics: The Noise/BW Trade-Off</td>
<td>12</td>
</tr>
<tr>
<td>Use with Operating Voltages Other than 3 V</td>
<td>13</td>
</tr>
<tr>
<td>Use as a Dual-Axis Tilt Sensor</td>
<td>13</td>
</tr>
<tr>
<td>Outline Dimensions</td>
<td>14</td>
</tr>
<tr>
<td>Ordering Guide</td>
<td>14</td>
</tr>
</tbody>
</table>

## REVISION HISTORY

6/05—Revision 0: Initial Version
## SPECIFICATIONS

$T_a = 25^\circ C$, $V_i = 3\, V$, $C_s = C_y = 0.1\, \mu F$, Acceleration = 0$ g, unless otherwise noted$^1$.

### Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSOR INPUT</td>
<td>Each axis</td>
<td>±2</td>
<td></td>
<td></td>
<td>g</td>
</tr>
<tr>
<td>Measurement Range</td>
<td>% of full scale</td>
<td>±0.2</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>Nonlinearity</td>
<td>X sensor to Y sensor</td>
<td>±0.1</td>
<td></td>
<td></td>
<td>Degrees</td>
</tr>
<tr>
<td>Alignment Error</td>
<td>Cross-Axis Sensitivity</td>
<td>±0.2</td>
<td></td>
<td></td>
<td>%</td>
</tr>
<tr>
<td>SENSITIVITY (RATIOMETRIC)$^2$</td>
<td>Each axis</td>
<td>378</td>
<td>420</td>
<td>462</td>
<td>mV/g</td>
</tr>
<tr>
<td>Sensitivity Change due to Temperature$^1$</td>
<td>$V_i = 3, V$</td>
<td>0.01</td>
<td></td>
<td></td>
<td>%/°C</td>
</tr>
<tr>
<td>ZERO g BIAS LEVEL (RATIOMETRIC)</td>
<td>$V_i = 3, V$</td>
<td>1.3</td>
<td>1.5</td>
<td>1.7</td>
<td>V</td>
</tr>
<tr>
<td>0 g Voltage at XOUT, YOUT</td>
<td>Initial 0 g Bias Deviation from Ideal</td>
<td>±50</td>
<td></td>
<td></td>
<td>mg</td>
</tr>
<tr>
<td>0 g Offset Vs. Temperature</td>
<td>&lt;$\pm 0.5$ mg/°C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOISE PERFORMANCE</td>
<td>at $25^\circ C$</td>
<td>220</td>
<td></td>
<td></td>
<td>µg/√Hz rms</td>
</tr>
<tr>
<td>Noise Density</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREQUENCY RESPONSE$^4$</td>
<td></td>
<td>0.002</td>
<td>10</td>
<td></td>
<td>µF</td>
</tr>
<tr>
<td>$C_s$, $C_x$ Range$^5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>% Tolerance</td>
<td></td>
<td>32 ± 15%</td>
<td></td>
<td></td>
<td>kΩ</td>
</tr>
<tr>
<td>4 RFLT Tolerance</td>
<td></td>
<td>5.5</td>
<td></td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td>Sensor Resonant Frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SELF-TEST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic Input Low</td>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Logic Input High</td>
<td></td>
<td>2.4</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>ST Input Resistance to Ground</td>
<td></td>
<td>50</td>
<td></td>
<td></td>
<td>kΩ</td>
</tr>
<tr>
<td>Output Change at XOUT, YOUT</td>
<td>Self-test 0 to 1</td>
<td>125</td>
<td></td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>OUTPUT AMPLIFIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Swing Low</td>
<td>No load</td>
<td>0.2</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Output Swing High</td>
<td>No load</td>
<td>2.7</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>POWER SUPPLY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating Voltage Range</td>
<td></td>
<td>2.4</td>
<td>6</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Quiescent Supply Current</td>
<td></td>
<td>0.45</td>
<td></td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Turn-On Time$^7$</td>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td>ms</td>
</tr>
<tr>
<td>TEMPERATURE</td>
<td>Operating Temperature Range</td>
<td>−20</td>
<td>70</td>
<td></td>
<td>°C</td>
</tr>
</tbody>
</table>

---

1. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.
2. Sensitivity is essentially ratiometric to $V_i$. For $V_i = 2.7\, V$ to $3.3\, V$, sensitivity is 138 mV/V/g to 142 mV/V/g typical.
3. Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
4. Actual frequency response controlled by user-supplied external capacitor ($C_x$, $C_y$).
5. $C_x$, $C_y$ increase turn-on time. Turn-on time is approximately $160 \times C_x$ or $C_y = 4\, ms$, where $C_x$, $C_y$ are in µF.
6. Self-test response changes cubically with $V_i$.
7. Larger values of $C_x$, $C_y$ increase turn-on time. Turn-on time is approximately $160 \times C_x$ or $C_y = 4\, ms$, where $C_x$, $C_y$ are in µF.
ABSOLUTE MAXIMUM RATINGS

Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration (Any Axis, Unpowered)</td>
<td>10,000 g</td>
</tr>
<tr>
<td>Acceleration (Any Axis, Powered)</td>
<td>10,000 g</td>
</tr>
<tr>
<td>$V_i$</td>
<td>$-0.3 \text{ V to } +7.0 \text{ V}$</td>
</tr>
<tr>
<td>All Other Pins</td>
<td>$\text{COM} - 0.3 \text{ V}$ to $\text{COM} + 0.3 \text{ V}$</td>
</tr>
<tr>
<td>Output Short-Circuit Duration</td>
<td>Indefinite</td>
</tr>
<tr>
<td>(Any Pin to Common)</td>
<td></td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>$-55^\circ\text{C to } +125^\circ\text{C}$</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>$-65^\circ\text{C to } +150^\circ\text{C}$</td>
</tr>
</tbody>
</table>

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although this product features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.
PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

Table 3. Pin Function Descriptions

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
<tr>
<td>2</td>
<td>ST</td>
<td>Self-Test</td>
</tr>
<tr>
<td>3</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
<tr>
<td>5</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>6</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>7</td>
<td>COM</td>
<td>Common</td>
</tr>
<tr>
<td>8</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
<tr>
<td>9</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
<tr>
<td>10</td>
<td>YOUT</td>
<td>Y-Channel Output</td>
</tr>
<tr>
<td>11</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
<tr>
<td>12</td>
<td>XOUT</td>
<td>X-Channel Output</td>
</tr>
<tr>
<td>13</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
<tr>
<td>14</td>
<td>Vsupply</td>
<td>2.4 V to 6 V</td>
</tr>
<tr>
<td>15</td>
<td>Vsupply</td>
<td>2.4 V to 6 V</td>
</tr>
<tr>
<td>16</td>
<td>NC</td>
<td>Do Not Connect</td>
</tr>
</tbody>
</table>
Figure 4. Recommended Soldering Profile

Table 4. Recommended Soldering Profile

<table>
<thead>
<tr>
<th>Profile Feature</th>
<th>Sn63/Pb37</th>
<th>Pb-Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Ramp Rate (T&lt;sub&gt;L&lt;/sub&gt; to T&lt;sub&gt;P&lt;/sub&gt;)</td>
<td>3°C/sec max</td>
<td>3°C/sec max</td>
</tr>
<tr>
<td>Preheat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum Temperature (T&lt;sub&gt;min&lt;/sub&gt;)</td>
<td>100°C</td>
<td>150°C</td>
</tr>
<tr>
<td>Minimum Temperature (T&lt;sub&gt;max&lt;/sub&gt;)</td>
<td>150°C</td>
<td>200°C</td>
</tr>
<tr>
<td>Time (T&lt;sub&gt;min&lt;/sub&gt; to T&lt;sub&gt;max&lt;/sub&gt;), t&lt;sub&gt;S&lt;/sub&gt;</td>
<td>60 sec – 120 sec</td>
<td>60 sec – 150 sec</td>
</tr>
<tr>
<td>T&lt;sub&gt;max&lt;/sub&gt; to T&lt;sub&gt;L&lt;/sub&gt; Ramp-Up Rate</td>
<td>3°C/sec</td>
<td>3°C/sec</td>
</tr>
<tr>
<td>Time Maintained Above Liquidous (T&lt;sub&gt;L&lt;/sub&gt;)</td>
<td>183°C</td>
<td>217°C</td>
</tr>
<tr>
<td>Liquidous Temperature (T&lt;sub&gt;L&lt;/sub&gt;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (t&lt;sub&gt;L&lt;/sub&gt;)</td>
<td>60 sec – 150 sec</td>
<td>60 sec – 150 sec</td>
</tr>
<tr>
<td>Peak Temperature (T&lt;sub&gt;P&lt;/sub&gt;)</td>
<td>240°C + 0°C/−5°C</td>
<td>260°C + 0°C/−5°C</td>
</tr>
<tr>
<td>Time within 5°C of Actual Peak Temperature (t&lt;sub&gt;P&lt;/sub&gt;)</td>
<td>10 sec – 30 sec</td>
<td>20 sec – 40 sec</td>
</tr>
<tr>
<td>Ramp-Down Rate</td>
<td>6°C/sec max</td>
<td>6°C/sec max</td>
</tr>
<tr>
<td>Time 25°C to Peak Temperature</td>
<td>6 min max</td>
<td>8 min max</td>
</tr>
</tbody>
</table>
TYPICAL PERFORMANCE CHARACTERISTICS ($V_S = 3.0\, V$)

**Figure 5.** X-Axis Zero g Bias at 25°C

**Figure 6.** X-Axis Zero g Bias Temperature Coefficient

**Figure 7.** X-Axis Sensitivity at 25°C

**Figure 8.** Y-Axis Zero g Bias at 25°C

**Figure 9.** Y-Axis Zero g Bias Temperature Coefficient

**Figure 10.** Y-Axis Sensitivity at 25°C
Figure 11. Zero g Bias vs. Temperature—Parts Soldered to PCB

Figure 12. X-Axis Noise Density at 25°C

Figure 13. Z vs. X Cross-Axis Sensitivity

Figure 14. Sensitivity vs. Temperature—Parts Soldered to PCB

Figure 15. Y-Axis Noise Density at 25°C

Figure 16. Z vs. Y Cross-Axis Sensitivity
Figure 17. X-Axis Self-Test Response at 25°C

Figure 18. Supply Current at 25°C

Figure 19. Y-Axis Self-Test Response at 25°C

Figure 20. Turn-On Time—Cx, Cy = 0.1 μF, Time Scale = 2 ms/DIV

Figure 21. Supply Current vs. Temperature Vs = 3V
Figure 22. Output Response vs. Orientation
THEORY OF OPERATION

The ADXL322 is a complete acceleration measurement system on a single monolithic IC. The ADXL322 has a measurement range of ±2 g. It contains a polysilicon surface micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. The accelerometer measures static acceleration forces, such as gravity, which allows it to be used as a tilt sensor.

The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and plates attached to the moving mass. The fixed plates are driven by 180° out-of-phase square waves. Acceleration deflects the beam and unbalances the differential capacitor, resulting in an output square wave whose amplitude is proportional to acceleration. Phase-sensitive demodulation techniques are then used to rectify the signal and determine the direction of the acceleration.

The demodulator’s output is amplified and brought off-chip through a 32 kΩ resistor. The user then sets the signal bandwidth of the device by adding a capacitor. This filtering improves measurement resolution and helps prevent aliasing.

PERFORMANCE

Rather than using additional temperature compensation circuitry, innovative design techniques were used to ensure built-in high performance. As a result, there is neither quantization error nor nonmonotonic behavior, and temperature hysteresis is very low (typically less than 5 mg over the −20°C to +70°C temperature range).

Figure 11 shows the zero g output performance of eight parts (X- and Y-axis) over a −20°C to +70°C temperature range.

Figure 14 demonstrates the typical sensitivity shift over temperature for supply voltages of 3 V. This is typically better than ±1% over the −20°C to +70°C temperature range.
APPLICATIONS

POWER SUPPLY DECOUPLING

For most applications, a single 0.1 µF capacitor, \( C_{DC} \), adequately decouples the accelerometer from noise on the power supply. However, in some cases, particularly where noise is present at the 140 kHz internal clock frequency (or any harmonic thereof), noise on the supply can cause interference on the ADXL322 output. If additional decoupling is needed, a 100 Ω (or smaller) resistor or ferrite bead can be inserted in the supply line. Additionally, a larger bulk bypass capacitor (in the 1 µF to 4.7 µF range) can be added in parallel to \( C_{DC} \).

SETTING THE BANDWIDTH USING \( C_x \) AND \( C_y \)

The ADXL322 has provisions for band-limiting the \( X_{OUT} \) and \( Y_{OUT} \) pins. Capacitors must be added at these pins to implement low-pass filtering for anti-aliasing and noise reduction. The equation for the 3 dB bandwidth is

\[
F_{-3\,\text{dB}} = \frac{1}{(2\pi \times 32\,\text{kΩ}) \times C_{(x,y)}}
\]

or more simply,

\[
F_{-3\,\text{dB}} = \frac{5}{C_{(x,y)}}
\]

The tolerance of the internal resistor (\( R_{FILT} \)) typically varies as much as ±15% of its nominal value (32 kΩ), and the bandwidth varies accordingly. A minimum capacitance of 2000 pF for \( C_x \) and \( C_y \) is required in all cases.

<table>
<thead>
<tr>
<th>Bandwidth (Hz)</th>
<th>Capacitor (µF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.7</td>
</tr>
<tr>
<td>10</td>
<td>0.47</td>
</tr>
<tr>
<td>50</td>
<td>0.10</td>
</tr>
<tr>
<td>100</td>
<td>0.05</td>
</tr>
<tr>
<td>200</td>
<td>0.027</td>
</tr>
<tr>
<td>500</td>
<td>0.01</td>
</tr>
</tbody>
</table>

SELF-TEST

The ST pin controls the self-test feature. When this pin is set to \( V_S \), an electrostatic force is exerted on the accelerometer beam. The resulting movement of the beam allows the user to test if the accelerometer is functional. The typical change in output is 300 mg (corresponding to 125 mV). This pin can be left open-circuit or connected to common (COM) in normal use.

The ST pin should never be exposed to voltages greater than \( V_S = 0.3 \) V. If this cannot be guaranteed due to the system design (for instance, if there are multiple supply voltages), then a low \( V_C \) clamping diode between ST and \( V_S \) is recommended.

DESIGN TRADE-OFFS FOR SELECTING FILTER CHARACTERISTICS: THE NOISE/BW TRADE-OFF

The accelerometer bandwidth selected ultimately determines the measurement resolution (smallest detectable acceleration). Filtering can be used to lower the noise floor, which improves the resolution of the accelerometer. Resolution is dependent on the analog filter bandwidth at \( X_{OUT} \) and \( Y_{OUT} \).

The output of the ADXL322 has a typical bandwidth of 2.5 kHz. To limit aliasing errors, the user must filter the signal at this point. The analog bandwidth must be no more than half the A/D sampling frequency to minimize aliasing. The analog bandwidth can be further decreased to reduce noise and improve resolution.

The ADXL322 noise has the characteristics of white Gaussian noise, which contributes equally at all frequencies and is described in terms of \( \mu g/\sqrt{\text{Hz}} \) (the noise is proportional to the square root of the accelerometer’s bandwidth). The user should limit bandwidth to the lowest frequency needed by the application in order to maximize the resolution and dynamic range of the accelerometer.

With the single-pole, roll-off characteristic, the typical noise of the ADXL322 is determined by

\[
\text{rmsNoise} = (220 \mu g/\sqrt{\text{Hz}}) \times (\sqrt{\text{BW} \times 1.6})
\]

At 100 Hz bandwidth the noise will be

\[
\text{rmsNoise} = (220 \mu g/\sqrt{\text{Hz}}) \times (\sqrt{100 \times 1.6}) = 2.8 \text{ mg}
\]

Often, the peak value of the noise is desired. Peak-to-peak noise can only be estimated by statistical methods. Table 6 is useful for estimating the probabilities of exceeding various peak values, given the rms value.

<table>
<thead>
<tr>
<th>Peak-to-Peak Value</th>
<th>% of Time That Noise Exceeds Nominal Peak-to-Peak Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 × rms</td>
<td>32</td>
</tr>
<tr>
<td>4 × rms</td>
<td>4.6</td>
</tr>
<tr>
<td>6 × rms</td>
<td>0.27</td>
</tr>
<tr>
<td>8 × rms</td>
<td>0.006</td>
</tr>
</tbody>
</table>

Table 6. Estimation of Peak-to-Peak Noise
Peak-to-peak noise values give the best estimate of the uncertainty in a single measurement. Table 7 gives the typical noise output of the ADXL322 for various $C_x$ and $C_y$ values.

<table>
<thead>
<tr>
<th>Bandwidth (Hz)</th>
<th>$C_x$, $C_y$ (µF)</th>
<th>RMS Noise (mg)</th>
<th>Peak-to-Peak Noise Estimate (mg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.47</td>
<td>0.9</td>
<td>5.3</td>
</tr>
<tr>
<td>50</td>
<td>0.1</td>
<td>2</td>
<td>11.8</td>
</tr>
<tr>
<td>100</td>
<td>0.047</td>
<td>2.8</td>
<td>16.7</td>
</tr>
<tr>
<td>500</td>
<td>0.01</td>
<td>6.2</td>
<td>37.3</td>
</tr>
</tbody>
</table>

**USE WITH OPERATING VOLTAGES OTHER THAN 3 V**

The ADXL322 is tested and specified at $V_s = 3$ V; however, this part can be powered with $V_s$ as low as 2.4 V or as high as 6 V. Note that some performance parameters change as the supply voltage is varied.

The ADXL322 output is ratiometric, so the output sensitivity (or scale factor) varies proportionally to supply voltage. At $V_s = 5$ V, the output sensitivity is typically 750 mV/g. At $V_s = 2.4$ V, the output sensitivity is typically 335 mV/g.

The zero g bias output is also ratiometric, so the zero g output is nominally equal to $V/2$ at all supply voltages.

The output noise is not ratiometric but is absolute in volts; therefore, the noise density decreases as the supply voltage increases. This is because the scale factor (mV/g) increases while the noise voltage remains constant. At $V_s = 5$ V, the noise density is typically 150 µg/√Hz, while at $V_s = 2.4$ V, the noise density is typically 300 µg/√Hz.

Self-test response in g is roughly proportional to the square of the supply voltage. However, when ratiometricity of sensitivity is factored in with supply voltage, the self-test response in volts is roughly proportional to the cube of the supply voltage. For example, at $V_s = 5$ V, the self-test response for the ADXL322 is approximately 610 mV. At $V_s = 2.4$ V, the self-test response is approximately 59 mV.

The supply current decreases as the supply voltage decreases. Typical current consumption at $V_s = 5$ V is 700 µA, and typical current consumption at $V_s = 2.4$ V is 340 µA.

**USE AS A DUAL-AXIS TILT SENSOR**

Tilt measurement is one of the ADXL322's most popular applications. An accelerometer uses the force of gravity as an input vector to determine the orientation of an object in space.

An accelerometer is most sensitive to tilt when its sensitive axis is perpendicular to the force of gravity (that is, when the package is parallel to the earth's surface). At this orientation, the accelerometer's sensitivity to changes in tilt is highest. When the accelerometer is oriented on axis to gravity (near its $+1$ g or $-1$ g reading), the change in output acceleration per degree of tilt is negligible. When the accelerometer is perpendicular to gravity, its output changes nearly 17.5 mg per degree of tilt. At 45°, its output changes at only 12.2 mg per degree of tilt, and resolution declines.

**Converting Acceleration to Tilt**

When the accelerometer is oriented so both its X-axis and Y-axis are parallel to the earth's surface, it can be used as a 2-axis tilt sensor with both a roll axis and a pitch axis. Once the output signal from the accelerometer has been converted to an acceleration that varies between $-1$ g and $+1$ g, the output tilt in degrees is calculated as

\[
PITCH = \text{ASIN}(A_x / 1\ g)\]

\[
ROLL = \text{ASIN}(A_y / 1\ g)\]

Be sure to account for overranges. It is possible for the accelerometers to output a signal greater than $\pm 1$ g due to vibration, shock, or other accelerations.
OUTLINE DIMENSIONS

Figure 23. 16-Lead Lead Frame Chip Scale Package [LFCSP_LQ]
4 mm × 4 mm Body, Thick Quad
(CP-16-5a*)
Dimensions shown in millimeters

ORDERING GUIDE

<table>
<thead>
<tr>
<th>Model</th>
<th>Measurement Range</th>
<th>Specified Voltage (V)</th>
<th>Temperature Range</th>
<th>Package Description</th>
<th>Package Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADXL322JCP</td>
<td>±2 g</td>
<td>3</td>
<td>−20°C to +70°C</td>
<td>16-Lead LFCSP_LQ</td>
<td>CP-16-5a</td>
</tr>
<tr>
<td>ADXL322JCP–REEL</td>
<td>±2 g</td>
<td>3</td>
<td>−20°C to +70°C</td>
<td>16-Lead LFCSP_LQ</td>
<td>CP-16-5a</td>
</tr>
<tr>
<td>ADXL322EB</td>
<td>±2.0 g</td>
<td>3</td>
<td>−20°C to +70°C</td>
<td>Evaluation Board</td>
<td></td>
</tr>
</tbody>
</table>

1 Lead finish—Matte tin.
Appendix D – Arduino Uno Datasheet

4.1 Arduino Uno

4.1.1 Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
• Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

4.1.2 Summary

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega328</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>14 (of which 6 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>6</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>32 KB (ATmega328) of which 0.5 KB used by bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>2 KB (ATmega328)</td>
</tr>
<tr>
<td>EEPROM</td>
<td>1 KB (ATmega328)</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
</tbody>
</table>

4.1.3 Schematic & Reference Design

EAGLE files: [arduino uno-Rev3-reference-design.zip](#)
Schematic: [arduino uno-Rev3-schematic.pdf](#)

Note: The Arduino reference design can use an Atmega8, 168, or 328. Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

4.1.4 Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board’s power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **GND.** Ground pins.

### 4.1.5 Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](https://www.arduino.cc/en/Reference/EEPROM)).

### 4.1.6 Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the `attachInterrupt()` function for details.

- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the `analogWrite()` function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the [SPI library](https://www.arduino.cc/en/Reference/SPI).

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the `analogReference()` function. Additionally, some pins have specialized functionality:

- **TWI**: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the [Wire library](https://www.arduino.cc/en/Reference/Wire).

There are a couple of other pins on the board:

- **AREF**: Reference voltage for the analog inputs. Used with `analogReference()`.
- **Reset**: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the [mapping between Arduino pins and ATmega328 ports](https://www.arduino.cc/en/Reference/BoardMap). The mapping for the Atmega8, 168, and 328 is identical.

### 4.1.7 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, **on Windows, a .inf file is required**. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](https://www.arduino.cc/en/Reference/SoftwareSerial) allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation](https://www.arduino.cc/en/Reference/Wire) for details. For SPI communication, use the [SPI library](https://www.arduino.cc/en/Reference/SPI).

### 4.1.8 Programming

The Arduino Uno can be programmed with the Arduino software ([download](https://www.arduino.cc/en/Main/Software)). Select "Arduino Uno from the Tools > Board menu (according to the microcontroller on your board). For details, see the [reference](https://www.arduino.cc/en/Reference) and [tutorials](https://www.arduino.cc/en/Tutorial).
The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details. The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel’s FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

### 4.1.9 Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.
4.1.10  **USB Overcurrent Protection**

The Arduino Uno has a resettable polyfuse that protects your computer’s USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

4.1.11  **Physical Characteristics**

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16”), not an even multiple of the 100 mil spacing of the other pins.
Appendix E - Hitec 422 Servo

ANNOUNCED SPECIFICATION OF HS-422 STANDARD DELUXE SERVO

1. TECHNICAL VALUES
   CONTROL SYSTEM: +PULSE WIDTH CONTROL 1500usec NEUTRAL
   OPERATING VOLTAGE RANGE: 4.8V TO 6.0V
   OPERATING TEMPERATURE RANGE: -20 TO +60°C
   TEST VOLTAGE: AT 4.8V
   OPERATING SPEED: 0.21sec/60° AT NO LOAD
   STALL TORQUE: 3.3kg.cm(45.82oz.in)
   OPERATING ANGLE: 45°ONE SIDE PULSE TRAVELING 400usec
   DIRECTION: CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec
   CURRENT DRAIN: 8mA/IDLE AND 150mA/NO LOAD RUNNING
   DEAD BAND WIDTH: 8usec
   CONNECTOR WIRE LENGTH: 300mm(11.81in)
   DIMENSIONS: 40.6x19.8x36.6mm(1.59x0.77x1.44in)
   WEIGHT: 4.1kg.cm(56.93oz.in)
   CONNECTOR WIRE LENGTH: 4.1kg.cm(56.93oz.in)
   DEATH BAND WIDTH: 0.16sec/60° AT NO LOAD
   STALL TORQUE: 4.1kg.cm(56.93oz.in)
   OPERATING ANGLE: 45°ONE SIDE PULSE TRAVELING 400usec
   DIRECTION: CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec
   CURRENT DRAIN: 8mA/IDLE AND 150mA/NO LOAD RUNNING
   DEAD BAND WIDTH: 8usec
   CONNECTOR WIRE LENGTH: 300mm(11.81in)
   DIMENSIONS: 40.6x19.8x36.6mm(1.59x0.77x1.44in)
   WEIGHT: 4.1kg.cm(56.93oz.in)

2. FEATURES
   3-POLE FERRITE MOTOR
   LONG LIFE POTENTIOMETER
   DUAL OILITE BUSHING
   INDIRECT POTENTIOMETER DRIVE

3. APPLICATIONS
   AIRCRAFT 20-60 SIZE
   30 SIZE HELICOPTERS
   STEERING AND THROTTLE SERVO FOR CARS
   TRUCK AND BOATS
References

[1] Auto-Leveling Platform – St. Mary’s University
http://engineering.stmarytx.edu/~nechon/
[2] Devry New Brunswick
http://www.youtube.com/watch?v=f9ALAvE3gBQ
[3] Self-leveling surface with arduino
http://www.youtube.com/watch?v=cTUBDagKdbA&feature=related
http://www.me.berkeley.edu/ME102/Past_Proj/f08/group_09/intro_objective.html
[5] Self-leveling platform
http://www.youtube.com/watch?v=TiTRUwU7kRs
[6] ECE572SelfLevelingPlatform
http://www.youtube.com/watch?v=CuN_ZkLK0gM&NR=1
[7] Stewart Platform University of Adger, Norway
http://www.youtube.com/watch?v=WmKnnp1xTPg&NR=1
[8] Self-leveling platform for control of autonomous lawn mower
[9] Proposed self-leveling controlled platform
http://www.youtube.com/watch?v=vmf1ThwrNM0
