Determining the position of the adiabatic demagnetization refrigerator and X-ray detector in the Micro-X sounding rocket payload

Kaitlyn Yoha
University of Florida REU participant, summer 2008
Under supervision of Dr. Tarek Saab

Abstract

The purpose of this study was to design and build a position alignment system that would monitor the position and orientation of the adiabatic demagnetization refrigerator (ADR) and detector front-end assembly inside the Micro-X sounding rocket payload. A testing prototype was designed and constructed in the lab to simulate the movement the ADR and the detector will experience in flight. This paper also discusses the method for determining the change in the ADR's position during the mission. Although the final position of the sensors on the ADR has not been determined, this study provides a basis for future study.

Introduction

The Micro-X is a sounding rocket payload (taken from the nautical term "to sound" meaning "to take measurements") that uses a high-energy-resolution X-ray microcalorimeter and an imaging mirror to observe X-ray spectra from an astronomical source [1]. Sounding rockets are useful for studying astronomical targets in the X-ray (and ultraviolet) portion of the electromagnetic spectrum because they exit the earth's atmosphere. The Micro-X mission will fly to an altitude of 160 km and take measurements for five minutes before coming back to earth [1].

The Micro-X mission is a new experiment, although it uses designs from previous X-ray astronomy missions [1]. The optical bench and X-ray mirror were used by the Supernova X-ray Spectrometer experiment that flew from Woomera, Australia in 1987, although the bench will be tailored to move the focal point to a new position that will be more appropriate for our experiment [1]. Also, the design for the Micro-X adiabatic demagnetization refrigerator (ADR) was based on the ADR used by the sounding rocket

payload X-ray Quantum Calorimeter that flew from the White Sands Missile Range in 1995, 1996, and 1999 [1]. It will be the first scientific observation with transition edge sensor (TES) microcalorimeters in space and will help develop future microcalorimeter systems for larger orbiting missions [1]. The Micro-X mission is set to launch in January 2011 from White Sands Missile Range in New Mexico [1].

The mission will examine the velocity structure and plasma conditions such as temperature, electron density, and ionization of the Bright Eastern Knot of the Puppis A supernova remnant [1]. The mission will also look for supernova ejecta (a star's material that is thrown outward into space by a supernova explosion) and measure bulk motions of the plasma [1]. The measurements we receive (in the form of high-resolution X-ray spectra) will be used to determine the temperature and ionization state of the gasses in Puppis A [1].

This high spectral resolution is achieved using arrays of microcalorimeters [1]. Photons are absorbed in the microcalorimeter and the energy is converted to heat. The resulting temperature rise is measured by the resistance change of the TES [1]. In order to do this, the microcalorimeters must be cooled to fifty mK through the use of an ADR [1].

The ADR works by thermally connecting a cylinder of salt called a "salt pill" to the X-ray detectors, then placing the salt pill in a strong magnetic field (salts are used because of their large magnetic moments) [2]. After the salt molecules have aligned with the external magnetic field, the strength of the magnetic field is decreased [2]. The thermal motion moves the molecules out of alignment, and the energy from the thermal motion is transformed into magnetic energy [2]. This process of controlling the magnetic

field is what cools the salt pill and the detectors. The cryogenic system that cools the device must be aligned with the optics that focus and image the incoming X-rays or it will not function properly.

Sensor and laser specifications

We are using four position-sensing diodes (PSD) to make sure the ADR is properly aligned with the TES in the rocket payload. The sensors are model DL100-7PCBA3 purchased from Pacific Silicon Sensor Incorporated; they are dual-axis position sensing diodes with sum and difference amplifiers [3]. Each sensor's active area is a 10-mm x 10-mm square that tracks the positions of a light spot on the surface of the photodiode and provides the voltages representing the X and Y positions [3].

As light strikes the diode, a current is generated at the center of the light power density [4]. A series of amplifiers convert these currents into voltages that can be read out by a voltmeter, oscilloscope, or a Virtual Instrument (VI) on LabVIEW [4].

The PSD responds to light wavelengths between 400 nm and 1100 nm [3]. It is also important that the light intensity should never be larger than 1.5W/cm^2 to avoid damaging the sensors [3]. The lasers we used are CL Series Circular Beam Laser Modules (Class 1, part no: CL5-0.4G-635) with optical output power of 0.4 mW and a minimum spot size of 60 μm at 10" distance [4]. We found the intensity of our lasers at the minimum spot size and maximum power using the equation

$$I = \frac{P}{A} \tag{1}$$

to be 14.147 W/cm^2. Using Eq. (1), we determined that the laser spot size should be no smaller than 0.185 mm in diameter to avoid damaging the sensor.

Although the ADR will reach temperatures close to absolute zero, the sensor's operable temperature range is 0 to 70 ^{o}C and the laser's operating temperature range is -10 to 50 ^{o}C [3][4]. This will not be a problem because the sensors are mounted on the outside of the ADR and will be working at the temperature of the rocket interior.

PSD calibration, Matlab and LabVIEW codes, and results

Our task was to calibrate the sensors and create a program that would determine the ADR's alignment in space when the mission is active. We carefully measured the CAD models of the ADR and bulkhead shown in fig. 1.

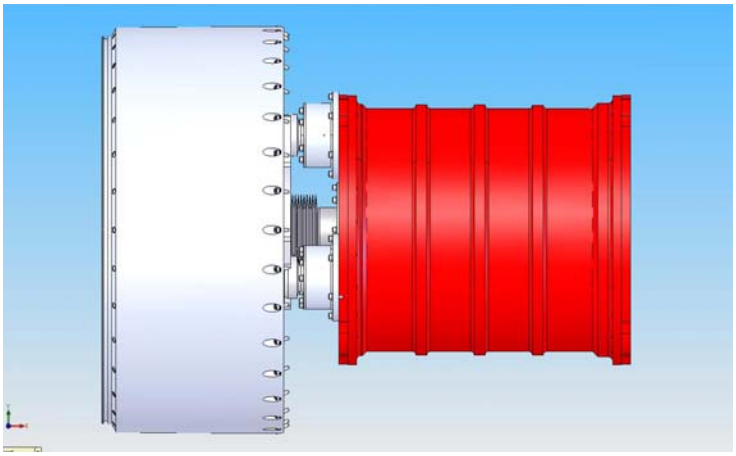


Figure 1: Front view of the ADR (shown in red) and Bulkhead (shown in white) [1].

From these measurements, we determined how much room there would be in the payload for our sensors and lasers. The bulkhead is hollow, so the lasers will be mounted on the inside, and the sensors will be attached to the aft end faceplate of the ADR facing the lasers. Originally we had planned for three sensors to be mounted on the faceplate facing the bulkhead, and one sensor mounted at a 90^{o} angle on the faceplate. The last sensor represents the Z-axis, as the sensors only measure in two dimensions. See Fig. 2

for clarification. These four sensors would provide enough information to determine the ADR's position after it has been allowed to rotate, tilt, and translate in three dimensions.

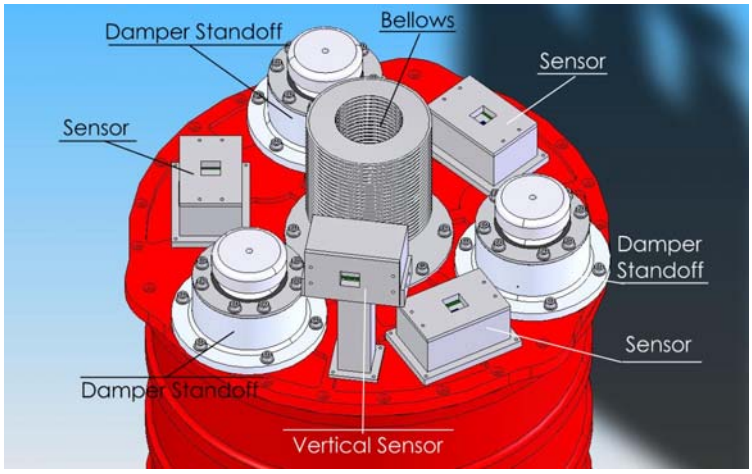


Figure 2: Original drawing of where the sensors will be located on the aft end faceplate of the ADR (bulkhead and some hardware has been removed for viewing purposes).

We first developed a program (see Appendix A for full code) on Matlab that calculated the change in position of the three sensors in the form of a translation (T) and an angle of rotation (Phi). This code did not calculate tilt and only worked in two dimensions, but it accurately described the movement of three sensors being translated and rotated.

This code works in tandem with a VI on LabVIEW (see Appendix B for full code); the VI reads the voltage output from the sensors and converts it to a numeric coordinate. These coordinates are then sent to Matlab where the computer finds the position of the centroid created by the initial coordinates of the three sensors on the faceplate and the position of the centroid created by the final coordinates. It then finds the vector between the two barycenters and outputs the components illustrating this linear movement. Then, the computer shifts both centroids to the origin. Once at the origin, all

coordinates are converted from Cartesian to polar coordinates and the final points are rotated until they overlap the initial points and the angle of rotation is displayed.

The Matlab program needed (as input) the ratio between the change in voltage produced by the sensor and the actual distance the laser moved. To find this ratio, we created a testing procedure of measuring the distance and voltages and applying a formula.

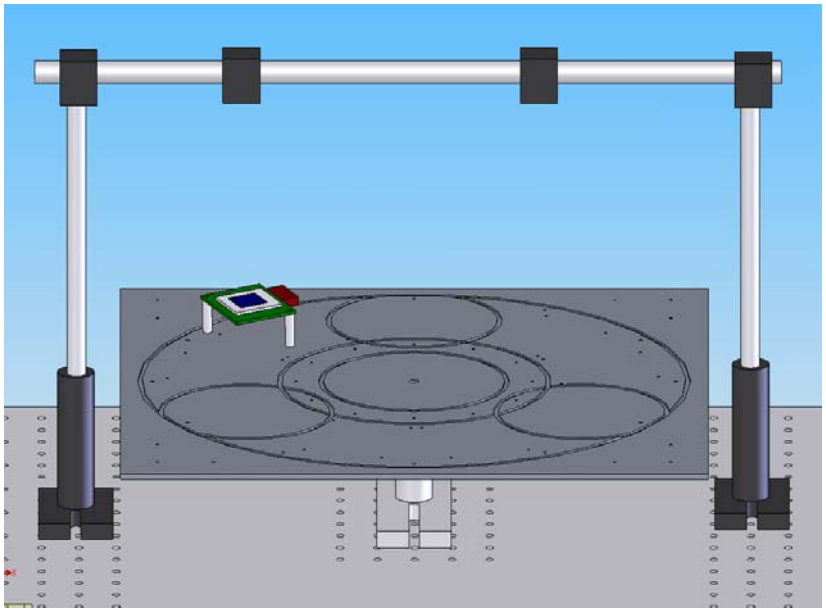


Figure 3: Drawing of the testing setup on SolidWorks.

The testing setup (See Fig. 3) we designed consisted of the sensors mounted to a prototype faceplate very similar to the aft end of the ADR. We then fastened the faceplate to two linear translation stages that were fixed perpendicularly to one another. A small post separated the faceplate from the stages so there was room to reach under the faceplate and adjust the position of the stages. We also had a rotary stage that could be mounted between the linear stages and the post and faceplate assembly, but the rotary stage was removed for most of our testing. Fig. 4 shows a photograph of the setup after the parts had been manufactured and assembled.

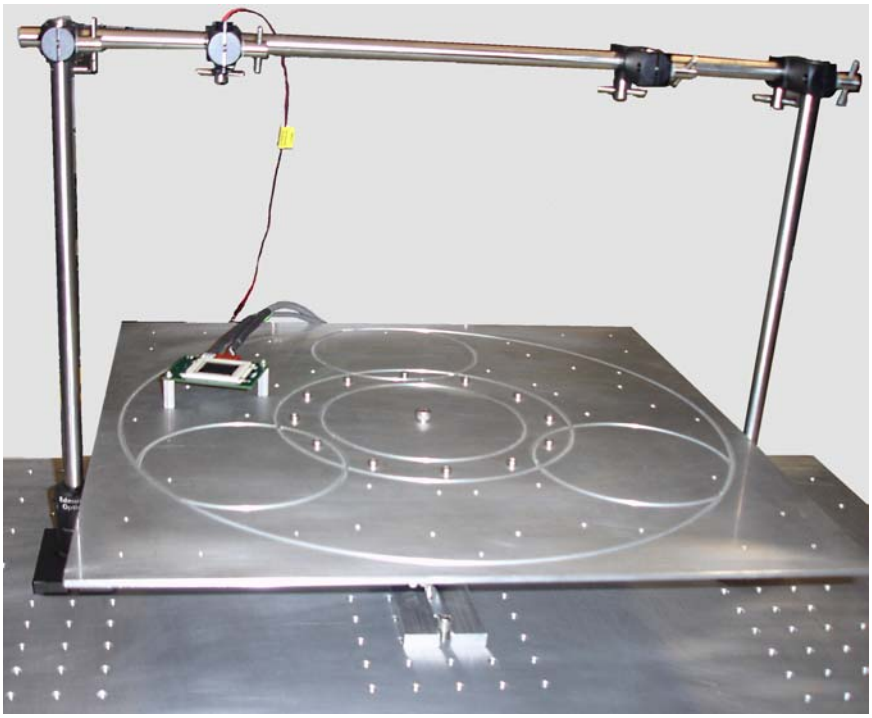


Figure 4: Photograph of the testing set up consisting of one sensor attached to the faceplate prototype with one laser mounted above the diode.

The sensor was wired into a breadboard that was interfaced with LabVIEW. The VI read the voltages outputted by the sensor when the laser was turned on and focused to an undamaging spot size; this position was recorded on the computer. The measurement taken was averaged over a period of two tenths of a second to eliminate some of the error caused by the tiny movement between the sensor and the laser. Then, we moved the laser, recorded the new voltage, and physically measured the distance the sensor moved with calipers.

The ratio between the actual change in position and change in voltage was determined using the following formula:

$$\frac{\sqrt{(\Delta d_x)^2 + (\Delta d_y)^2}}{\sqrt{(V_{X2_{av}} - V_{X1_{av}})^2 + (V_{Y2_{av}} - V_{Y1_{av}})^2}} = \mathrm{R} \qquad (2)$$

where d is the actual change in position and V is the average voltage recorded by the VI. The numerical answer found using Eq. (2) tells us that for every volt recorded by the VI, the sensor moves R mm. In our tests, R was 34.65317 (see Appendix C for a sample of the data). However, error still exists in that number because the calipers we used only measured to two decimal places, and it was difficult to actually measure the change in position. More tests are necessary to confirm this R-value as being the most accurate.

This process was repeated for X-axis translations, Y-axis translations, and both X and Y translations until there was a very small standard deviation between the trials. We used the ratio found through these trials to convert the voltage readouts into position coordinates for the computer to decipher.

We conducted the calibration procedure mainly in the dark because ambient light generates errors by creating a current at the center of the diode. This current increases the overall current and reduces the resolution from the laser itself [3]. The lights, however, were turned on to measure the change in position with the calipers.

Our overall goal was to create a procedure that would correctly identify the change in motion in three dimensions. After we had successfully interpreted between the voltages from the sensors and the change in position for just an X and Y translation, we started working on a code that incorporated all three dimensions.

This new code (see Appendix D for partial code) is similar to the old code and works by inputting the positions from the VI and converting them to spherical coordinates. From the spherical coordinates, the computer can determine the angles Theta (angle the ADR tilts away from the Z-axis) and Phi (angle the ADR rotates about the Z-axis) and the linear translations between the initial and final position of the ADR.

This code does not work; before we could finish it, we determined that there was a more efficient way to calculate the orientation of the sensors and the ADR.

The new design has two sensors facing the bulkhead and two sensors at a 90^{o} angle to the faceplate to measure the Z-axis as shown in Fig. 5. More testing is necessary to vindicate this change in sensor position, but in theory, this adjustment will eliminate some of the uncertainty in the orientation because there will be two sensors displaying the vertical movement instead of one. Coupled with the two sensors showing the X and Y positions, this is enough information to describe all movement in any direction.

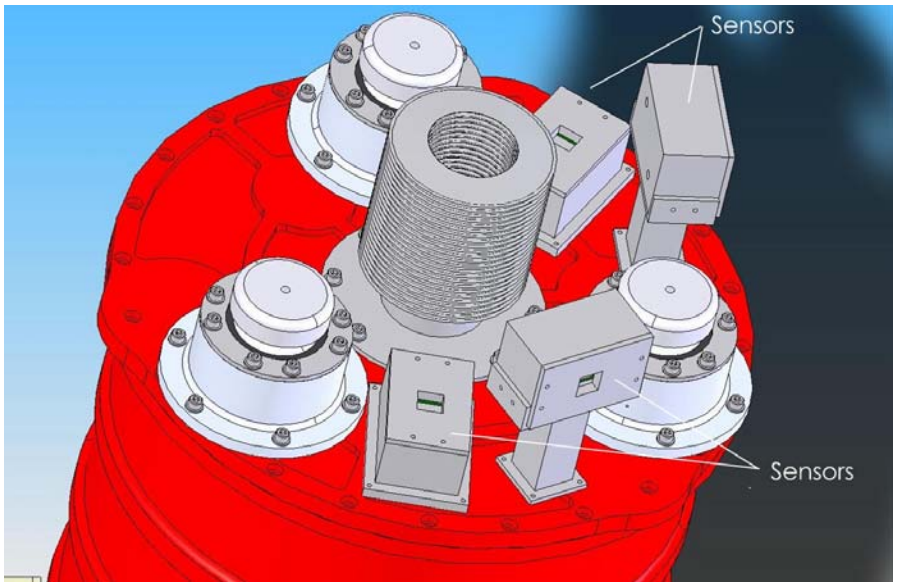


Figure 5: Drawing of the final position of the sensors on the ADR faceplate.

Although we did not have a chance to test the tilt in a laboratory situation, we did create test points by hand that represented what the VI would input into Matlab. We hope that once our program is completed, we will be successful in deciphering the movement of the four sensors in three dimensions using these test points. Then the program will be tested with the actual sensors in the lab to verify their accuracy.

We also did not utilize the rotary stage for testing purposes, although we have several sets of test data points (created by hand) that replicated the test points the VI would have inputted into Matlab given the same parameters. Matlab consistently interpreted the movement correctly, and given more time to work, we would have tested the rotation in the lab as well.

Sensor housing unit

The next issue to solve involved keeping the fragile sensors protected while inside the rocket. We designed a small metal housing for the sensors that will protect them throughout the assembly, transportation, and flight. The sensors will be attached firmly to the housing unit, and the houses will be attached to the faceplate of the ADR, or to a standoff that holds the unit at a 90^{o} angle and is elevated to be inside the bulkhead.

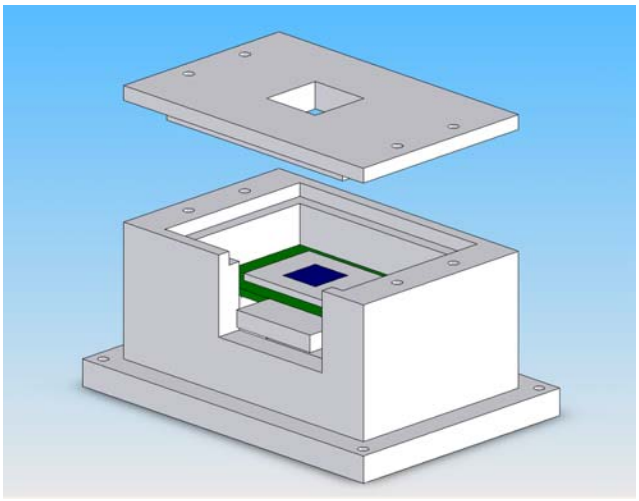


Figure 6: Drawing of the housing unit for the sensors. Sensor location is shown. Lid pictured is the one used for taking data.

The houses are aluminum and have interchangeable lids for when the rocket is being assembled and transported to the launch site, and for when the sensor is going to be in use during the flight. The first lid is solid and completely encases the sensor. The

second lid has a square opening slightly larger than the sensor that will allow the laser beam to strike the diode during flight. The lids are easy to change and are firmly attached to the base of the housing with screws. The house can be observed in Fig. 6.

Conclusion

Although we did not finish our testing, we have provided the groundwork for further study. We designed and manufactured the equipment for the testing procedure that will allow the sensors to be standardized and usable for the Micro-X mission. In the short time we had, we were able to calculate the change in position of the sensors based on the voltage readouts through the programs we created on LabVIEW and Matlab. Once the codes are finalized and the testing procedure completed, we should be able to determine the sensors, and thus the ADR's, position and orientation during the flight.

Acknowledgements

I would like to thank Dr. Tarek Saab for advising me on this project, Tylor Whitmer for his invaluable programming skills and help throughout this session, and Patrick Wikus for the SolidWorks drawing of the ADR and the data package on the Micro-X mission. I would also like to specially thank Mr. Michael J. Yoha for his extensive knowledge of electronics, mechanics, programming, and for knowing exactly how everything works. In addition, thanks to Dr. Durdana Balakishiyeva and the following graduate students: Gendith Sardane, Rene Molina, and Avtendil Achelashvili, for their help throughout the project. I would also like to thank the staff in the Machine Shop, especially Mr. Bill Malphurs and Mr. Marc Link, for their expertise in machining,

and tolerating my constant badgering in the shop. Also, thanks to Dr. Kevin Ingersent, REU director, and Kristin Nichola, program assistant. Special thanks to the NSF for funding, and the University of Florida for hosting the REU.

References

[1] P. Wikus, Mission Number 36.245 Data Package for the Micro-X Mission Initiation Conference (MIC) Revision E, 2008.

[2] NASA's Imagine the Universe! How Does an Adiabatic Demagnetization Refrigerator work? http://imagine.gsfc.nasa.gov/docs/teachers/lessons/xray_spectra/background-adr.html.

[3] Data Sheet: Dual Axis PSD Sum and Difference Amplifier Specifications, http://www.pacific-sensor.com/pdf/DL100-7PCBA3.pdf.

[4] CL Series Circular Beam Laser Module, http://wstech.koredesign.com/pdf/CL5-0.4G-635.pdf.

Appendix A

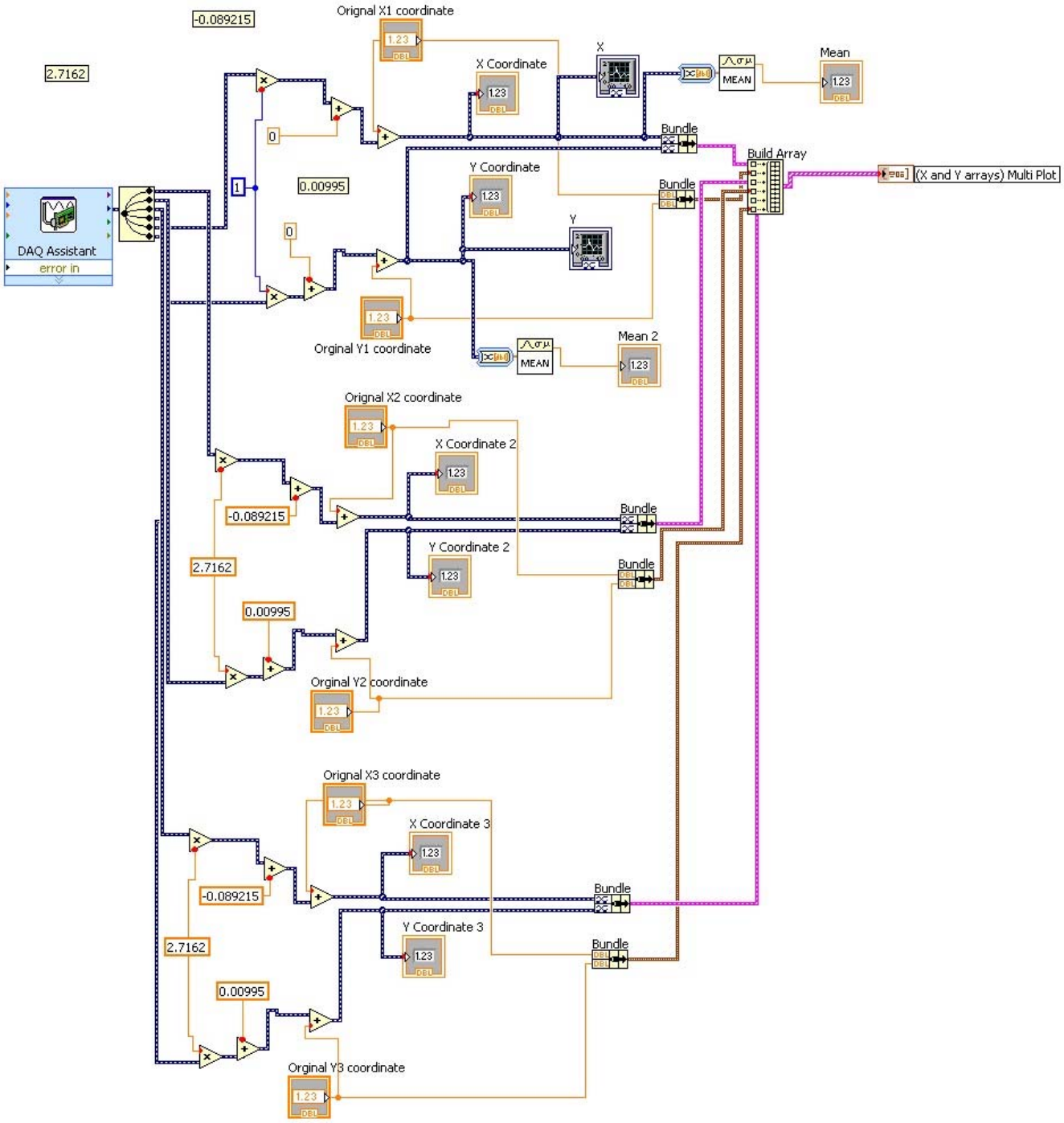
Original Matlab Code for finding T and Phi only.
Written by T. Whitmer, and K. Yoha.

```
%   This program takes in Initial Points, and Change in Distances, performs a series of
%   calculations, and outputs Phi and T(components describing the linear translations)
%
%   Variable Dictionary:
%      vX = original x, y coordinates for each of the three sensors on top of the ADR.
%      D1 = Vector of how much the diode moved. This is what the LabVIEW VI finds.
%      T = Linear translation the sensors experience
%      Phi = angle the sensors were rotated

function [T, Phi] = InverseTransformADR(vX1 ,vX2 ,vX3, D1, D2, D3)

%  vXpp = Final (transitional, and rotational) x and y coordinates for each sensor
vX1pp = vX1 + D1;
vX2pp = vX2 + D2;
vX3pp = vX3 + D3;

%%%%%%%%%%%%%%%%%
%  TRANSLATION SECTION  %
%%%%%%%%%%%%%%%%%

%  COM = Center of Mass
PostT_COM = (vX3pp +vX2pp + vX1pp) ./3; % After rotation/translation
PreT_COM =  (vX1 + vX2 + vX3) ./3; % before rotation/Translation (original points)

%  Changing original points Center of Mass to the Origin
TempVX1 = vX1 - PreT_COM;
TempVX2 = vX2 - PreT_COM;
TempVX3 = vX3 - PreT_COM;

%  Subtract the Center of Mass from the final and the initial to get the translation
T = PostT_COM - PreT_COM;

%%%%%%%%%%%%%%%%%
%   ROTATION SECTION     %
%%%%%%%%%%%%%%%%%

%  Subtract the PostT_COM from the Final points to gain the (vX1p) post rotational
%  points and pre translation points

vX1p = vX1pp - PostT_COM;
vX2p = vX2pp - PostT_COM;
vX3p = vX3pp - PostT_COM;
```

```
%  NOTE: polar Coordinates = (Phi, Magnitude)
%  Finding Polar Coordinates for TempOrginal points (COM at origin)
%  Pol1 = polar coordinates of TempvX1
[Pol1(1), Pol1(2)] =cart2pol(TempVX1(1), TempVX1(2));
[Pol2(1), Pol2(2)] =cart2pol(TempVX2(1), TempVX2(2));
[Pol3(1), Pol3(2)] =cart2pol(TempVX3(1), TempVX3(2));

%  Finding Polar Coordinates for prime points (after translation)
[Pol1p(1), Pol1p(2)] =cart2pol(vX1p(1), vX1p(2));
[Pol2p(1), Pol2p(2)] =cart2pol(vX2p(1), vX2p(2));
[Pol3p(1), Pol3p(2)] =cart2pol(vX3p(1), vX3p(2));

%  Angle rotated is the Post rotational angle - the pre Rotational Angle
Phi1 = Pol1p(1) - Pol1(1);
Phi2 = Pol2p(1) - Pol2(1);
Phi3 = Pol3p(1) - Pol3(1);

%%%%%%%%%%
%   Error Handling   %
%%%%%%%%%%

% Condition: Two Phi angles aren't equal
%     True: 1) Display Error message, telling the two Phi's used
%               2) Display Error by subtracting the two Phi's used to compare
%     False: Nothing

if Phi1 ~= Phi2
   disp('There is an error between  angle 1 and 2')
   Error = Phi1 - Phi2
end

if Phi1 ~= Phi3
   disp('There is an error between angle 1 and 3')
   Error = Phi1 - Phi3
end

if Phi2 ~= Phi3
   disp('There is an error between angle 2 and 3')
   Error = Phi3 - Phi2
end

%  Takes the Average of the Three Angles to get Phi
Phi = (Phi1 + Phi2 + Phi3)./3;

%  END
```

Appendix B

Block Diagram of the VI on LabVIEW



Appendix C

Data taken from linear translation (X direction only)

Distance input	50.76	48.95	42.48	45.06	47.58	51.99	46.05
Voltage X	0.0699897	0.022532	-0.174907	-0.0898008	-0.0191169	0.100232	-0.0579085
Voltage y	-0.0202241	-0.00517178	0.058053	0.0270753	0.00694084	-0.0311108	0.0208611
Voltage Final	0.022532	-0.0318594	-0.0725126	-0.0333662	0.0165596	0.0122685	-0.00098276
Voltage Final	-0.00517178	0.0121608	0.0252404	0.0124083	-0.00403118	-0.00216579	0.00184155
Final Distance	48.95	47.2	45.66	47.08	48.91	48.71	48.24
Ratio	36.35442987	30.65535488	29.57495978	34.64279079	35.63241348	35.41986322	36.48843174

Distance input	44.09	45.34	47.82	50.85	53.64	51.11	47.65
Voltage X	-0.114333	-0.0781112	-0.0113854	0.0684032	0.151022	0.0766921	-0.0152168
Voltage y	0.0379028	0.0275011	0.00512934	-0.0199933	-0.047204	-0.0237284	0.00615454
Voltage Final	-0.057178	-0.0340643	0.0564294	0.130853	0.0862455	0.0497622	-0.0631461
Voltage Final	0.0201459	0.0132089	-0.0171041	-0.0400605	-0.0268264	-0.0136757	0.0216088
Final Distance	46.13	46.99	50.43	53.06	51.47	50.14	45.9
Ratio	34.08531284	35.63127897	36.57179656	33.69172542	31.95588123	33.74497327	34.75033341

Distance input	48.21	45.1	49.06	46.57	49.78	48.27
Voltage X	-0.00144386	-0.0842576	0.0206423	-0.0438933	0.0400052	0.000780582
Voltage y	0.00144482	0.028554	-0.00514746	0.015255	-0.0115957	0.0017252
Voltage Final	-0.0636058	-0.00971794	-0.02885	-0.103174	0.00695038	-0.04619
Voltage Final	0.0222368	0.00528622	0.0111146	0.0342798	4.39E-05	0.0172987
Final Distance	45.87	47.91	47.17	44.46	48.51	46.5
Ratio	35.6995577	35.98558913	36.27951685	33.89085154	36.2398785	35.76838363

Appendix D

Non-computing Matlab Code
Written by T. Whitmer and K. Yoha.

```
%  This program inputs the initial X and Y positions from the four sensors (three in a
%  triangle measuring the horizontal change, and one measuring the vertical change) and
%  the vector describing the change in position from each sensor, performs a series of
%  calculations, and outputs T (the linear translation in X, Y, and Z), Phi (the angle the
%  sensors rotated about the vertical Z-axis), and Theta (the angle the sensors rotated
%  about a line in the X-Y plane).
%
%  Variable Dictionary
%     vX = Original X, Y coordinates of each sensor.  Sensor 4 is the one that measures
%             the vertical change.
%     D1 = Change in position relative to the sensor specified.  This is read by LabVIEW.
%     T = Linear translation in X, Y, and Z.
%     Phi = Angle the sensors rotated about the vertical Z-axis.
%     Theta = Angle the sensors rotated about a line in the X-Y plane.

function [T, Phi, Theta] = InverseTransformADRTESTing(vX1 ,vX2 ,vX3,vX4,
          D1,D2,D3,D4)

%  vX1pp = Final (transitional, and rotational) X, Y coordinates (for each sensor
%  specified)
vX1pp = vX1 + D1;
vX2pp = vX2 + D2;
vX3pp = vX3 + D3;

%%%%%%%%%%%%%%%%%%%%%%%%
%       Finding the Z value the final position    %
%%%%%%%%%%%%%%%%%%%%%%%%

% Finding the distance between the original points
Dis1 = distance([vX1(1),0],[0,vX1(2)]);
Dis2 = distance([vX2(1),0],[0,vX2(2)]);
Dis3 = distance([vX3(1),0],[0,vX3(2)]);

DisD1 = distance([D1(1),0], [0,D1(2)]);
DisD2 = distance([D2(1),0], [0,D2(2)]);
DisD3 = distance([D3(1),0], [0,D3(2)]);

TriangleLeg1 = Dis1 - DisD1;
TriangleLeg2 = Dis2 - DisD2;
TriangleLeg3 = Dis3 - DisD3;

Height1 = (Dis1^2 - TriangleLeg1^2)^.5;
```

```
Height2 = (Dis2^2 - TriangleLeg2^2)^.5;
Height3 = (Dis3^2 - TriangleLeg3^2)^.5;

if Dis1 < TriangleLeg1
   Z1 = Height1 * -1;
else
   Z1 = Height1;
end

if Dis2 < TriangleLeg2
   Z2 = Height2 * -1;
else
   Z2 = Height2;
end

if Dis3 < TriangleLeg3
   Z3 = Height3 * -1;
else
   Z3 = Height3;
end

%  vX1ppp = the X, Y, and Z position of each sensor after being rotated, translated, and
%  tilted.
vX1ppp = [vX1pp(1), vX1pp(2), Z1];
vX2ppp = [vX2pp(1), vX2pp(2), Z2];
vX3ppp = [vX3pp(1), vX3pp(2), Z3];

%  Testing the rest of program.
vX1ppp = [vX1pp(1), vX1pp(2), .68674256814];
vX2ppp = [vX2pp(1), vX2pp(2), .064838009679];
vX3ppp = [vX3pp(1), vX3pp(2), .064838009679];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Horizontal and Vertical Translation Section     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  If all the sensor are still in the X-Y plane,
%  TRUE: Vertical translations = the y value of the diode

   PostT_COM = (vX1ppp +vX2ppp + vX3ppp) ./3; % After rotation and translation
   PreT_COM =  (vX1 + vX2 + vX3) ./3; %  before rotation and translation (original
  %  points)

   % if (Dis12pp == Dis12) && (Dis13pp == Dis13) && (Dis23pp == Dis23)
  if PostT_COM == PreT_COM
    T = [0,0,D4(2)];
```

```
else
    %  Moving the original Center of Mass to the Origin
    TempVX1 = vX1 - PreT_COM;
    TempVX2 = vX2 - PreT_COM;
    TempVX3 = vX3 - PreT_COM;

    %  Subtract the final Center of Mass from the initial to get the translation
    T = PostT_COM - PreT_COM;
end

%  Moving the final coordinates to the origin
TempVX1ppp = vX1ppp - PostT_COM;
TempVX2ppp = vX2ppp - PostT_COM;
TempVX3ppp = vX3ppp - PostT_COM;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      Cartesian Coordinates to Spherical Coordinates    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  [Phi (X, Y), Theta (Z), R] - our standards
%  [Theta (X, Y), Phi (Z) , R] - Computer
[Spherical1ppp(1), Spherical1ppp(2), Spherical1ppp(3)] =
cart2sph(TempVX1ppp(1),TempVX1ppp(2),TempVX1ppp(3));
[Spherical2ppp(1), Spherical2ppp(2), Spherical2ppp(3)] =
cart2sph(TempVX2ppp(1),TempVX2ppp(2),TempVX2ppp(3));
[Spherical3ppp(1), Spherical3ppp(2), Spherical3ppp(3)] =
cart2sph(TempVX3ppp(1),TempVX3ppp(2),TempVX3ppp(3));

[Spherical1(1), Spherical1(2), Spherical1(3)] =
cart2sph(TempVX1(1),TempVX1(2),TempVX1(3));
[Spherical2(1), Spherical2(2), Spherical2(3)] =
cart2sph(TempVX2(1),TempVX2(2),TempVX2(3));
[Spherical3(1), Spherical3(2), Spherical3(3)] =
cart2sph(TempVX3(1),TempVX3(2),TempVX3(3));

Spherical1
Spherical2
Spherical3

Theta1 = Spherical1ppp(2) - Spherical1(2);
Theta2 = Spherical2ppp(2) - Spherical2(2);
Theta3 = Spherical3ppp(2) - Spherical3(2);

%  Averaging Theta values
Theta = (Theta1 + Theta2 + Theta3) /3;
```

```
%  Spherical to Cartesian (3d to 2d)
Spherical1ppp(1) = Spherical1(1);
Spherical2ppp(1) = Spherical2(1);
Spherical3ppp(1) = Spherical3(1);

[vX1pp(1), vX1pp(2) , vX1pp(3)] = sph2cart(Spherical1ppp(1), Spherical1ppp(2),
Spherical1ppp(3));
[vX2pp(1), vX2pp(2) , vX2pp(3)] = sph2cart(Spherical2ppp(1), Spherical2ppp(2),
Spherical2ppp(3));
[vX3pp(1), vX3pp(2) , vX3pp(3)] = sph2cart(Spherical3ppp(1), Spherical3ppp(2),
Spherical3ppp(3));

%%%%%%%%%%%%%%%%%%%
%       ROTATION SECTION          %
%%%%%%%%%%%%%%%%%%%

%  Subtracting the PostT_COM from the Final points to gain the (vX1p) post rotational
%  points and pre translation points
vX1p = vX1pp - PostT_COM;
vX2p = vX2pp - PostT_COM;
vX3p = vX3pp - PostT_COM;

%  NOTE: polar Coordinates = (Phi, Magnitude)
%  Finding Polar Coordinates for TempOrginal points (COM at origin)
%  Pol1 = polar coordinates of TempvX1
[Pol1(1), Pol1(2)] =cart2pol(TempVX1(1), TempVX1(2));
[Pol2(1), Pol2(2)] =cart2pol(TempVX2(1), TempVX2(2));
[Pol3(1), Pol3(2)] =cart2pol(TempVX3(1), TempVX3(2));

%  Finding Polar Coordinates for prime points (after translation)
[Pol1p(1), Pol1p(2)] =cart2pol(vX1p(1), vX1p(2));
[Pol2p(1), Pol2p(2)] =cart2pol(vX2p(1), vX2p(2));
[Pol3p(1), Pol3p(2)] =cart2pol(vX3p(1), vX3p(2));

%  Angles rotated is the Post rotational angle  - the pre Rotational Angle
Phi1 = Pol1p(1) - Pol1(1);
Phi2 = Pol2p(1) - Pol2(1);
Phi3 = Pol3p(1) - Pol3(1);

%%%%%%%%%%%
%    Error Handling    %
%%%%%%%%%%%

%   Condition:  Two Phi angles aren't equal
%       True: 1) Display Error message, telling the two Phi's used
%             2) Display Error by subtracting the two Phi's used to compare
```

```
%     False: Nothing

if Phi1 ~= Phi2
    disp('There is an error between  angle 1 and 2');
    Error = Phi1 - Phi2;
end

if Phi1 ~= Phi3
    disp('There is an error between angle 1 and 3');
    Error = Phi1 - Phi3;
end

if Phi2 ~= Phi3
    disp('There is an error between angle 2 and 3');
    Error = Phi3 - Phi2;
end

%  Takes the Average of the Three Angles to get Phi
Phi= (Phi1 + Phi2 + Phi3)./3;

T
Phi
Theta

%  END
```