IGV ACTUATION CONTROL SYSTEM INTEGRATION

Design Review #4 (Final)

Major Design Experience/Design and Manufacture III

Prepared by
Huang Yihe
Shan Zhengdong
Wu Jiayi
Yu Zhaolin
Zhu Qinzhou
Undergraduate Engineers from Group 18

Instructed by
Dr. Kai Xu
Dr. Chengbin Ma

Sponsored by
Lau, Ow and Pan, Wen Yong
Siemens Limited China

May 14, 2020

***Abstract*—This is a project sponsored by SIEMENS about IGV (inlet guide vane) actuation control system integration, which, in other words, considers the control of inlet air of a gas turbine. This should be achieved through an adjustable vane system. The purpose of this project is to design a system involving the mechanism of turbine and its control method to fulfill the goal. This report covers most materials in the report for the first three design review; the final design and the way it is selected among all the concepts are added. The progress status and QFD are also presented as their final version.**

***Keywords*—gas turbine, inlet guide vane, Siemens, HMI, PLC.**

CONTENTS

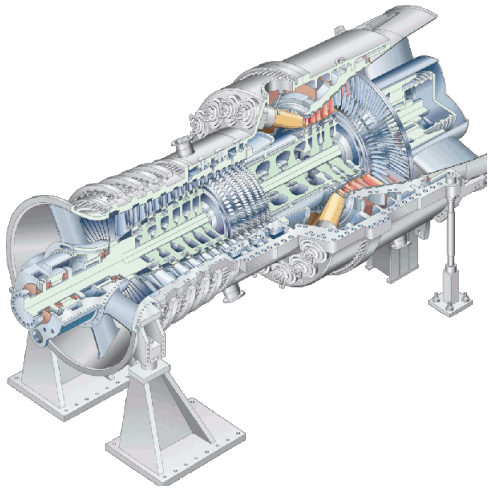


Fig. 1. Gas turbine structure

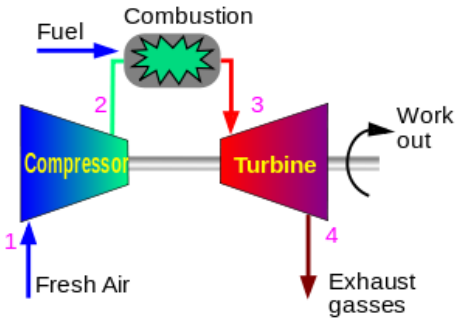


Fig. 2. Concise control diagram

I. Introduction

The combustion turbine is a machine which can obtain mechanical energy from the high temperature and high pressure combusted air. It is usually used as a motor or engine. (Figure 1)

By controlling the IGV — inlet guide vanes, which locate at the entrance of the compressor, the certain amount of fresh air can be imported into the system. (Figure 2) By controlling the inlet air, the amount of energy used can be managed within a proper range in order for energy saving. The project can also be embedded in transportation area, for turbine engines are the key part for planes or vessels.

This IGV project is serial. Now is its fourth stage. The target of the pervious design is to control the angle of vane by the rotation of the ring. The ring and the axis of the vane is connected by a linkage (Figure 3) so that when the ring moved, the conjunction moves at the tangential direction, which drive the linkage and change the angel of the vane.

After the vane change the angel, getting each vanes' angle is the main part of the project. The pervious team managed a method to calculate the angle of each vane.

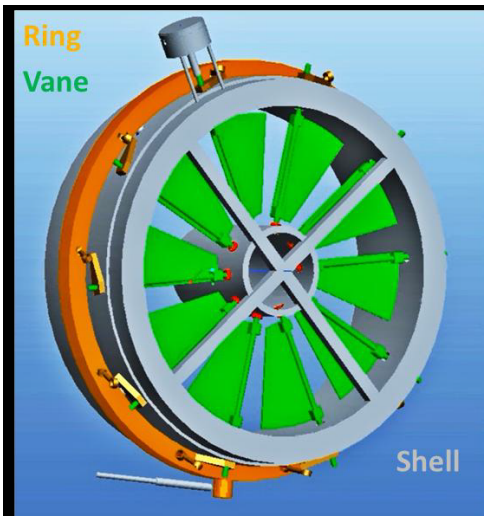


Fig. 3. Design of previous project

TABLE I. Litereature Surey Detail

Database	Google Scholar, IEEE
Key words	IGV control gas turbine vane actuator PI control for gas turbine
Total documents	7
Most cited author	G.S. Stone and P.M. Schoonmaker (17 times)
Published year	6 from 2000 to 2008, 1 from 1990
Most successful institute	Mitsubishi (2 papers)
Most successful country U.S.	(3 papers)

There are two main problem in the previous design and need to be solved. The first one is that during the rotation of the ring, the deformation changed not only in the tangential and radial direction, but also at axial direction, which cause the miscalculation and many error. The second one is that although the previous team showed a method which can calculate the angle of each vane by the mathematical method and data from sensor. The result cannot be verified very clearly.

II. Literature Survey

For the literature survey, Google Scholar and IEEE were used as the major database. Seven documents were found after the key words searching. As Table I shows, G.S. Stone and P.M. Schoonmakers paper was cited by others seventeen times. Six papers were published from 2000 to 2008, one from 1990. Mitsubishi and U.S. turned out to be the most successful institute and nation in this field.

Until now, all the researches being found relevant to the project focus on the control of the vane angle and the design of actuator involving PI-control. The accuracy and precision of the system still has space for improvement.

III. Quantification of Engineering Specifications

A. Customer Requirements

After discussions with the company sponsors and faculty advisors, the following customer requirements have been identified:

1) Compatible with existing turbine models.

The IGV actuation and control solution be adopted in future SIEMENS products, therefore our design should work with existing SIEMENS compressor turbines. Since the prototype should be based on work done by previous teams working on the same project, sticking with the basic layout of the previous prototypes and making evolutionary tweaks and improvements ensure that it meet this particular requirement.

2) Accurate and precise vane angle control.
A maximum vane angle deviation among all vanes under steady state of 1.8 degrees is required. The company sponsors also assume accurate angle control as one of the basic requirements.
3) Durable.
The actuation mechanism must be durable. It is required that the mechanism should withstand 5500 cycles of operation (from full-open to close).
4) Inexpensive.
The company sponsors expect a reasonable budget to develop the solution. The budget of the whole project is limited at a maximum amount of 7000 RMB.
5) Responsive.
It is expected that a response time of the entire control-actuation system to be no greater than 2 seconds. This means that the actuation scheme shall be flexible and responsive in addition to being accurate and precise.
6) User-firendly.
A more user-friendly human-machine interface (HMI) is a critical section in the project. The HMI should give the user the option to generate various waveforms of control inputs.
7) Efficient use of sensors.
Efficient use of sensors prevents the system from measuring the vane angles directly as inputs to our control system. This decision is not only due to the high cost of the sensors, but also because of the harsh working conditions of these sensors. Maintaining a minimum amount of sensors guarantees the reliability of the system.
8) Clean and compact assembly.
The final prototype should by presented in a neat and clean way. The company sponsors expect the quality and presentation of our prototype to be comparable to those of a proof-of-the-concept product.

B. Engineering Specifications

Starting from the Customer Requirements identified in the previous section, we quantify them into the following Engineering Specifications. Some of them are already quantified by our company sponsors in their requirements.

1) Yield strengths of the materials involved in building the system.
Yield strengths of the casing and linkage materials affect directly the reliability of the actuation mechanism as well as the durability of our design.
2) Friction factor among the linkage mechanism.
The frictions among linkages in the actuation mechanism could impede the fast response of the system as well as introduce unintended wear and tear to mechanical structure, shortening its service life. Friction factor determines the magnitude of friction force among those linkage structures, and all contacting positions shall be carefully designed to minimize the friction factor. Like the previous one, the specific value of this specification is to be determined later.
3) Effective spring constant.
Since the casing of the compressor turbine is subject to significant thermal expansion during operation, the project will be using the spring-loaded mechanism developed by previous teams to hold the actuation ring in position. The springs involved plays an important role in absorbing the thermal strain and preventing the overload of the ring, but should also retain some degree of stiffness to maintain the precision of vane angle control. The desired spring constant value is also to be determined later.
4) Vane angle deviation.
Vane angle deviation quantifies the precision of the IVG actuation and control mechanism. As specified by the company sponsors, the maximum allowed vane angle deviation among all vanes is 1.8 degrees.
5) Cost.
The total cost of our prototype either justifies or falsifies its merit for production use. The goal is keep the cost below the budget, which is7000 RMB in total.
6) Control system response time.
The overall response time of the control-actuation system quantitatively measures the responsiveness of the design. The company sponsors require that the overall response time to be no greater than 2 seconds.
7) Size of the container case.
The overall size of the container case of the prototype (not to be confused with the turbine casing, which refers to the cylindrical wall of the compressor turbine) is an important measurement of the compactness of the presentation of the design. After the sizes of individual components were finalized, the case was determined to be $1.5\text{m}\times0.8\text{m}\times0.8\text{m}$.

C. Quality Function Development

After the requirements and specifications were determined, a Quality Function Development was formed to decide which requirements or specifications are more crucial for this project, like what is shown in Figure 4. Binary comparison method were selected to calculate the weight of all requirements. Then, for each requirements, one must find out whether the specifications have a strong connection with the requirement. “9” means strong connection, while “1” refers to the weakest connection. Next, for each specification, the total value are calculated by timing the weight and connection together. The previous prototype were chosen to be the benchmarks. The three most important specifications were highlighted.

IV. Concept Generation

Before the design was finalized, multiple conceptual solutions were generated and were subject to design decisions. In order to generate conceptual solutions to the stated problem,

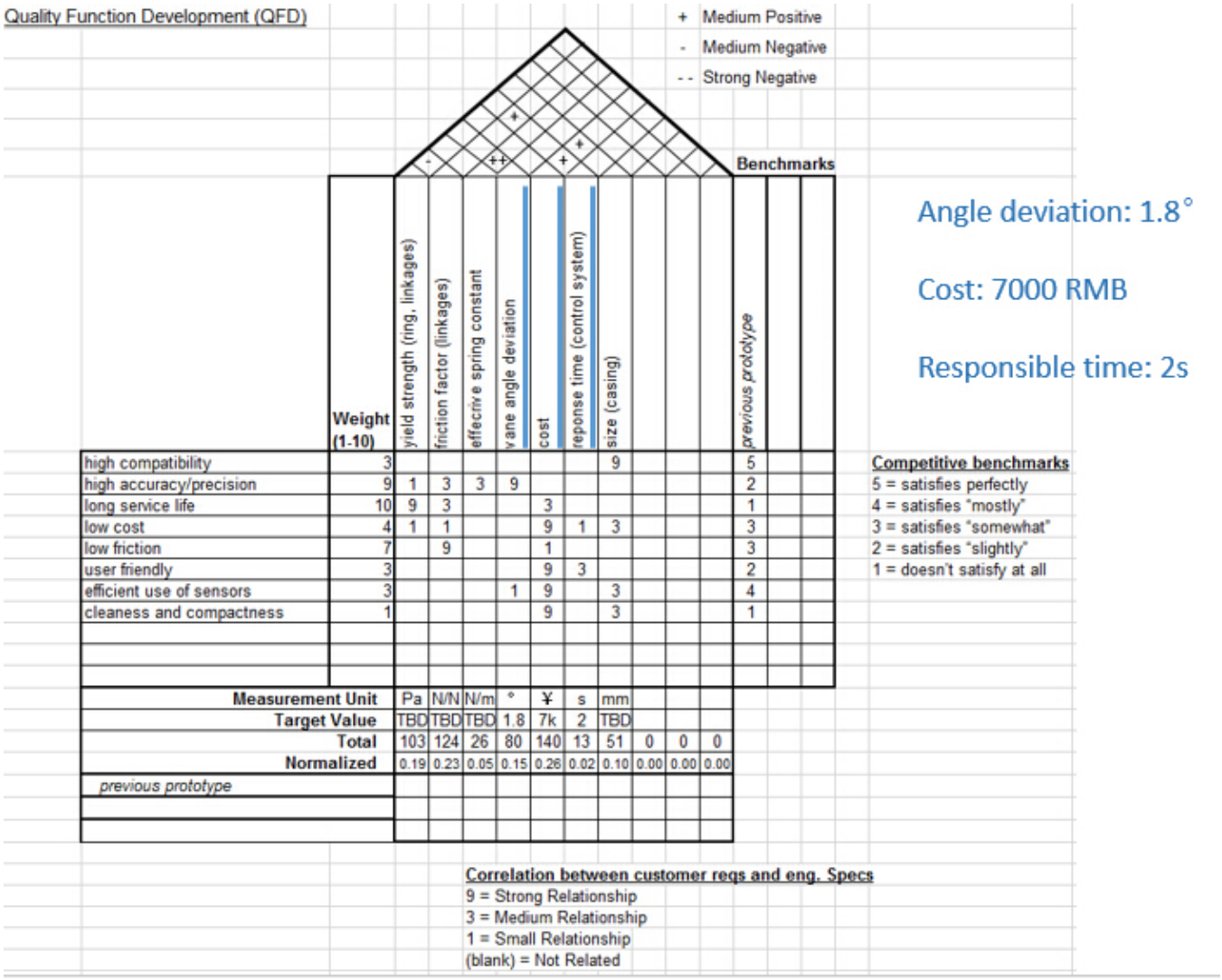


Fig. 4. Quality Function Development (QFD)

a functional decomposition that was applicable to all potential solutions must be established. Literature reviews and examinations of existing similar solutions justified the following decomposition scheme: transmission, support, sensing, computation, control, and human-machine interaction. Figure 5 shows the functional decompostion.

Concepts were generated using the method of brainstorming with pre-determined design criteria. All concepts, provided that they adhered to at least one pre-determined criterion, were accepted as valid. Judgements and design decisions were not made until the concept selection process. Each concept category mentioned above will be described in the rest of the section.

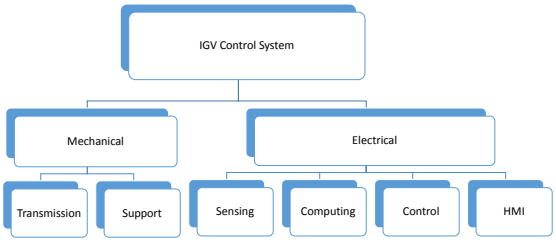


Fig. 5. Functional decomposition chart

A. *Transmission*

Transmission refers to the mechanism that transforms the rotation of the actuation ring into the individual rotations of the vanes along their shafts. The design of the actuation ring was preserved because of its prevalence and success in various related industrial designs.

Concepts regarding the transmission mechanisms were generated with three main considerations in mind: rigidity, flexibility, and cost. Rigidity describes the strengths of constrains that are met by moving components along all unwanted directions (positively related). Flexibility describes the amount of frictions or deformations between or within moving components under normal working conditions (negatively related).

Costs incurred during both manufacturing and maintainence processes account for the third criterion.

For example, three conceptual solutions were proposed for the concept category of transmission: a gearwheel-based solution, a slider-based solution, and a spherical-joint-based (eye bolt linkages) one. The gearwheel-based one had very high rigidity but potential high costs as well, and the eye-bolt-based one performs better in terms of flexibility but did not provide the same level of rigidity as gearwheels did. Detailed descriptions of the concepts involved are to be found in section A.

B. Support

Support refers to the contact between the actuation ring and the shell of the turbine inlet. Support mechanisms must provide firm and reliable mounting points for the actuation ring without compromising the ring's ability to rotate or translate in the desired directions. Undesired movements of the ring should be kept minimum at the meantime, if not completely eliminated. Detailed descriptions of the concepts involved are to be found in section A.

C. Sensing

One major objective of this project is to design a minimum sensing scheme that provides accurate measurements/estimations of all vane angles. The measurements/estimations were to be utilized by the programmable logic controller (PLC) to perform real-time calculations and controls.

Based on the assumed mechanical properties of the final design, it was revealed that the offset of the actuation ring due to actuator activity (eccentricity) and the rotational angle of the actuation ring were the only two variable physical quantities that needed to be measured in real-time. All other quantities (mostly dimensional quantities) should be determined during the design/manufacturing process and would not vary under normal working conditions.

Concepts under this category explored different types of sensors that might be incorporated to the final design, especially in terms of their performance, cost, and compatibility. Performance includes both precision and linearity, cost mainly addresses retail prices, and compatibility concerns both the ease of assembly and the type of electrical signals generated. Detailed descriptions of the concepts involved are to be found in section A.

D. Computation

Since the design process did not explore the option of directly measuring all vane angles by the corresponding sensors, the final design must possess certain computation capabilities to apply the math model transforming limited sensor inputs into quantities of interest, such as individual and average vane angles. Due to limitation of resources, the computation is to be performed by the PLC as an extra payload in addition to its original responsibility of carrying out the control algorithm. This part was largely design-specific and was one of the most flexible functional components in the entire solution. For this reason, no concepts in this category were generated before the sensing and all other mechanical concepts were finalized.

TABLE II. SCORING MATRIX FOR SELECTING CONCEPTS OF TRANSMISSION

	Weight	Gear	Slider	Eye bolt
Accuracy/Precision	*9*	5	3	3
Service Life	*10*	2	3	4
Cost-effectiveness	*4*	1	5	4
Flexibility	*7*	1	2	5
Total (weighted)		76	91	<u>118</u>

Detailed descriptions of the implemented computation scheme are to be found in the section VII.

E. Control

Feedback control was one of the required features of this project, and mature theories had been established in the field to provide us with control solutions that suited the stated problem. Standard PI control was implemented for the project. No other concepts were explored during the design process to save time and resources.

F. Human-machine Interaction

A user-friendly Human-machine Interface (HMI) was requested by the sponsors of the project. It was also a very flexible component of the final design, and therefore it was not subject to design decisions during concept generation. Detailed descriptions of the implemented HMI are to be found in subsection VIII-A.

V. CONCEPT SELECTION

Scoring matrices were used to determine the winner concept in each concept categories. The final design concept included the following winner concepts: eye bolt linkages for transmission, a solution based on linear bearings and rollers for support, and a combination of resistor gauge meters and angle transducers for sensing. Concept categories not mentioned here were not subject to concept selections as per discussions in the Concept Generation section.

While the detailed scoring matrices and corresponding criteria for comparisons are to be presented in Appendix H-B, the rest of this section describes how comparisons were made in general using the example of selecting the winning concept for transmission.

As mentioned in the previous section, three solutions were proposed for transmission. The criteria based on which comparisons were made were then extracted from the engineering specifications. For the concept of transmission, the applicable engineering specifications are accuracy/precision, durability, cost-effectiveness, and flexibility. The team discussed the pros and cons of each proposed concept and assigned scores to each concept for each criterion. The final score of each concept was the weighted average of its scores in all criterion. The weight involved in calculation was consistent with the weight derived during the quantification of engineering specifications.

Table II shows the scoring matrix of the concept category of transmission. The selected concept (eye bolt linkages) won the contest because of its cost-effectiveness and flexibility, but it did not provide the accuracy or precision that can be achieved by a more costly solution such as the gear-based one. However, it was determined that in this particular application,

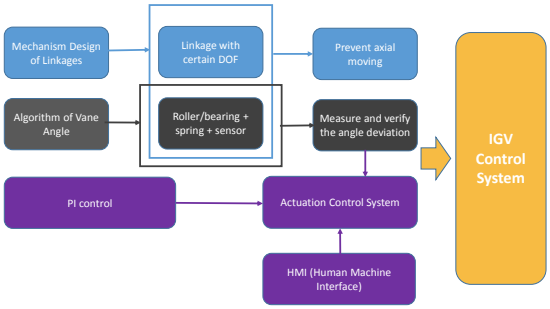


Fig. 6. Concept flow chart

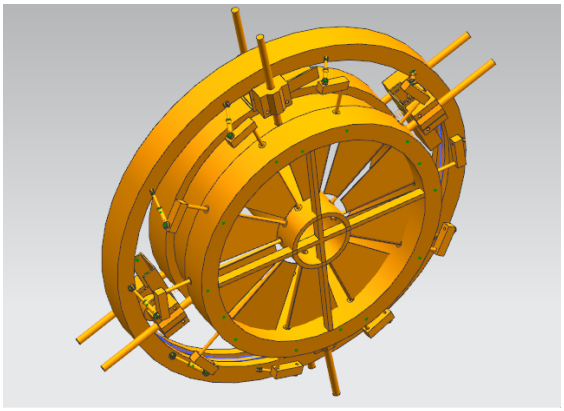


Fig. 7. Chosen concept of the mechanical subsystem

maintenance costs and durability outweigh the rigidity of the system, which can be compensated by proper sensing and control systems. This justifies the selected concept.

VI. Concept Description

The selected concepts form an integrated solution that addresses the stated problem. The concept flow chart of the entire solution is shown in Figure 6.

The mechanical subsystem provides necessary constrains as well as power transmission. Support mechanisms ensure that the actuation ring can freely rotate, or even translate within its own plane, in response to actuator inputs, but repress the axial movement of the ring. Support mechanisms also provide necessary mounting points for the sensors. Transmission mechanisms convert the rotation of the ring into individual rotations of the vanes, and meanwhile ensure smooth operation even if the actuation ring is subject to minor displacement (eccentricity) due to actuator inputs. Figure 7 depicts the chosen mechanical concepts, and Figure 8 gives a closer look at the chosen concept for support.

The electrical subsystem carries out computation, control, and monitoring, and provides an interface between the system and human operators. The PLC interprets sensor signals, processes the gathered data, consolidates and organizes them to make control decisions and to present to system administrators. The electrical subsystem also includes the actuator, a PLC-controlled AC motor, which delivers mechanical power to the system.

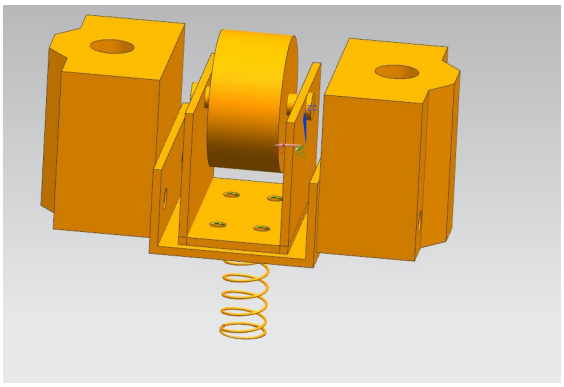


Fig. 8. Chosen concept of the support mechanism

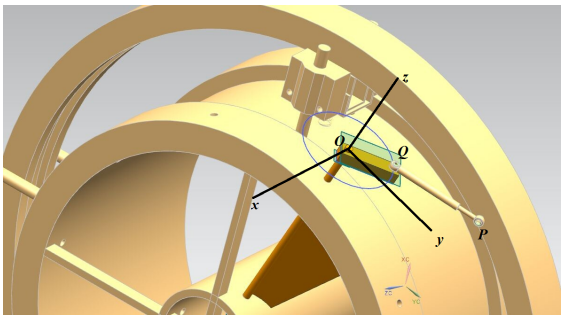


Fig. 9. Coordinate system set-up for a certain vane

VII. Analysis

This section describes the math model, the algorithms, and other technical details of the final design. The math model predicts the performance of the system, the algorithms implement the math model in the context of available sensor inputs and are applied to the PLC, and extra attentions must be paid to address computational limitations and timing requirements.

A. Math Model

The math model of the final design transforms the two identified variable quantities, the offset (eccentricity) and rotation of the actuation ring, into all 10 individual vane angles. To ease the computation, we assume an independent Cartesian coordinate system for each vane. The detailed set-up of the coordinate system is shown in figure 9.

Please note that the x and y axes of the coordinate system are elevated for a better presentation layout. The actual x axis used for calculations lies on the central axis of the turbine inlet shell, and the y axis should also be adjusted accordingly to the position that goes through the central axis while being perpendicular to both the central axis and the vane shaft (z axis).

To find the vane angle, the mechanical structures involved must be abstracted into basic geometric shapes. The abstracted system is shown in figure 10 with critical dimensional and variable quantities labeled. Points P and Q are the two ends of the eye bolt linkage, and dimension constraints restrict point P's movement to a circle with radius r, which represents the effective length of the fixed-end linkage (a.k.a. the length of segment OP in figure 9). Point Q, on the other hand, is

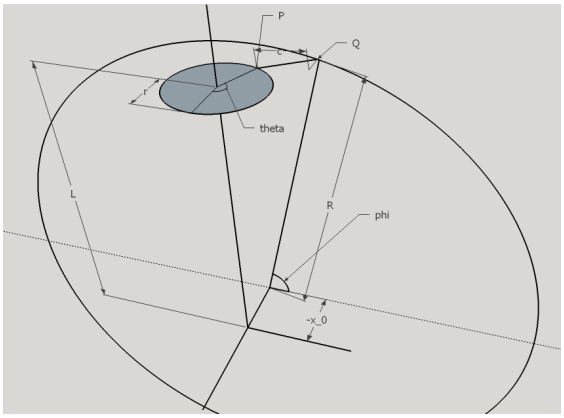


Fig. 10. Abstraction of the mechanical system

fixed on the actuation ring, and is therefore restricted to the circumference of the circle with radius R, where R represents the effective radius of the actuation ring. The vane angle is represented by angle θ (theta) in figure 10, and angle ϕ (phi) in the same figure directly relates to the rotational angle of the ring itself.

Other dimensional quantities involved in the derivation of the math model are: x_0: effective distance between the vane shaft and the actuation ring (negative value); L: the effective elevated height of the fixed-end linkage; c: the effect length of the eye bolt linkage.

Since points P and Q move on circles, their coordinates in the established coordinate system can be parametrized using trigonometry functions and angles θ and ϕ.

$$P : (r\cos\theta, r\sin\theta, L)$$

$$Q : (x_0, R\cos\phi, R\sin\phi)$$

Until now the derivation process does not take into account the effect of ring offset (eccentricity). To take eccentricity into account, adjustments need to be made to the y and z coordinates of point Q by adding the projected displacements along the $+y$ and $+z$ directions. Let Δy and Δz denote these displacements, the revised coordinates for points P and Q becomes:

$$P : (r\cos\theta, r\sin\theta, L)$$

$$Q : (x_0, R\cos\phi + \Delta y, R\sin\phi + \Delta z)$$

The equation governing the relationship between vane angle and variable quantities (ring eccentricity and rotation) is derived from the dimensional constraint that the distant between points P and Q are fixed to the length of the eye bolt linkage (c in figure 10). Equation 1 relates individual vane angle (θ) to variable quantities (Δy, Δz, and ϕ).

$$\begin{aligned}(x_0 - r\cos\theta)^2 &+ (R\cos\phi + \Delta y - r\sin\theta)^2 \\ &+ (R\sin\phi + \Delta z - L)^2 = c^2 \end{aligned} \quad (1)$$

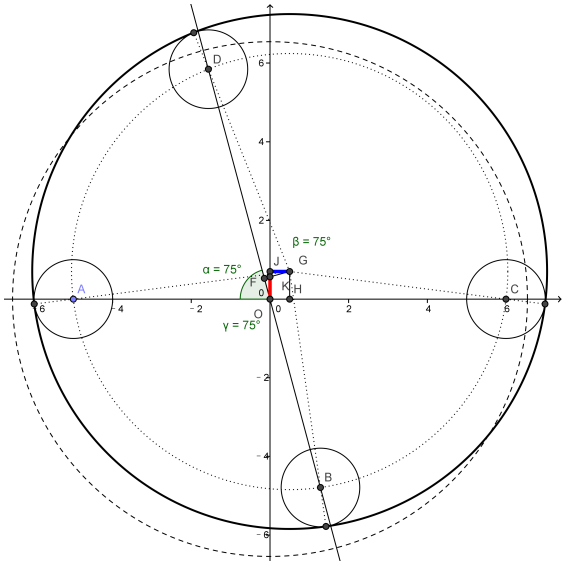


Fig. 11. Eccentricity of the ring and displacements of support mechanisms

The math model derived here applies to an individual vane, and separate calculation is required to derive the angle of each vane. It should be noted that Δy and Δz will be different for different vanes under a given ring eccentricity condition, because of coordinate system rotations, but ϕ shall remain common for all vanes at any given time instant due to the nature of rotation.

B. Algorithms

The algorithms described in this subsection relate sensor inputs to the quantities that are present in the math model. The algorithms are responsible for two distinct tasks, to be completed in series: 1. Derivation of variable quantities from sensor inputs; 2. Derivation of vane angles from variable quantities. As per conventions of this report, variable quantities refer to Δy and Δz in each coordinate system as well as the universal ring rotation angle ϕ.

According to the final design, sensor inputs consist of four resistor gauge meter inputs and one angle transducer input. The four resistor gauge meters are mounted on the four support mechanisms, and the angle transducer is mounted on a selected vane shaft. Resistor gauge meter inputs are used to calculate the eccentricity of the ring, and the angle transducer directly measures one vane angle, and based on which the ring rotation angle ϕ can be calculated from the math model.

Figure 11 examines the relation between the displacements of support mechanisms and the eccentricity of the ring. Support mechanisms are not orthogonally placed due to interference with the transmission mechanisms; they were placed at an angle α as shown in the figure. The displacements of support mechanisms are effectively measured by resistor gauge meters mounted on them. In the figure, the dashed circle shows the ring with zero eccentricity, and the solid circle shows the ring's position with 2-D offsets due to actuation inputs. Point G is the center of the ring after deformation, and small circles centered at A, B, C, and D represent the rollers on the four support mechanisms.

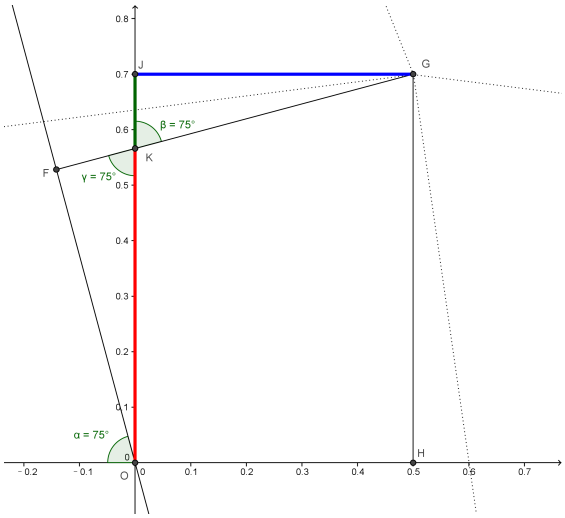


Fig. 12. Finding the y coordinate of point G

Note that the figure was not drawn to actual scale – the rollers are much smaller than their sizes shown in the figure; they were being exaggerated for better illustration of geometric relations.

The first task to be completed by the algorithms is to compute the position of G in the coordinate system shown in the figure from effective sensor inputs $|AO|$, $|BO|$, $|CO|$, and $|DO|$.

The main geometric constraint in figure 11 states that circle G (the solid big circle) is the circumscribed circle of circles A, B, C, and D. Since circles A, B, C, and D are have identical radii, the distances between point G to points A, B, C, and D are also identical to each other. Hence, this constraint results in isosceles triangles $\triangle GAC$ and $\triangle GDB$. Therefore the x coordinate of G can be calculated by halving the difference of two sensor inputs:

$$G_x = \frac{|CO| - |AO|}{2}$$

Calculation of the y coordinate of G is a little trickier because of the non-90° value of angle α. Figure 12 shows an enlarged portion of figure 11 that shows the derivation process of y coordinate of point G.

In the enlarged figure, GF was made perpendicular to OD (point D was out of scope of the enlarged figure). It can be clearly seen in the figure that the y coordinate of G consists of two segments: segment OK marked in red and length KJ marked in dark green. Trigonometry functions relate these quantities to known values by their geometric definitions directly:

$$OK = \frac{OF}{\sin\alpha},\ KJ = \frac{JG}{\tan\alpha}$$

where

$$OF = \frac{|DO| - |BO|}{2},\ JG = G_x$$

TABLE III. LATENCIES OF SELECTED S7-200 STL FLOATING POINT (REAL) INSTRUCTIONS

Instruction	Description	Max Latency (μs)
*R	Floating point multiplication	166
/R	Floating point division	230
+R	Floating point addition	99
-R	Floating point subtraction	100
SQRT	Calculate square root	550
SIN	Calculate Sine	1070
COS	Calculate Cosine	1070
TAN	Calculate Tangent	1300

Hence the y coordinate of point G is the arithmetic sum of OK and KJ

$$G_y = \frac{|DO| - |BO|}{2\sin\alpha} + \frac{G_x}{\tan\alpha}$$

Equation 2 summarizes the derivation of eccentricity of the ring from sensor inputs and concludes the first task of the algorithms.

$$\begin{aligned} G_x &= \frac{|CO| - |AO|}{2} \\ G_y &= \frac{|DO| - |BO|}{2\sin\alpha} + \frac{G_x}{\tan\alpha} \end{aligned} \quad (2)$$

The second task of the algorithms determines the universal rotational angle ϕ. ϕ is not being monitored by any of the angle transducers due to the lack of reliable mounting points, but is calculated from a known vane angle (θ_0), which is being monitored by the angle transducer mounted on a selected vane shaft. With knowledge of the offsets of the ring and vane angle θ_0, the math model described by equation 1 can be applied to find ϕ. The derived ϕ can then be used to calculate the rest of the vane angles.

C. Implementation Notes

The programmable logic controller (PLC) is not designed to perform complex mathematical calculations, therefore special attentions need to be paid when implementing the algorithms in the form of PLC programs. The S7-200 PLC System Manual supplied by Siemens [8] describes the computation capability of the product in terms of instruction execution times. Table III summaries the computational performance of key instructions.

General assessments of the computation capability of the PLC show that the PLC performs computation at a much lower speed than most commercial computers. For example, Intel's Pentium 4 microprocessor computes a full-precision Sine value of any given input within 200 CPU cycles [9] (less than 67 ns, or more than 15,000 times faster than the PLC). Given the limited computation capability of the PLC, it is not feasible to directly solve equation 1 to find θ and ϕ.

Instead of resorting to inverse trigonometry functions or solving transcendental equations, the method of Taylor series expansion was applied to approximate the math model described by Equation 1 in the PLC program. All Sine and

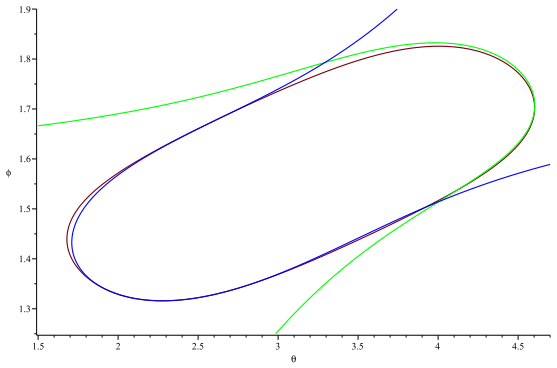


Fig. 13. Approximation of the math model for PLC implementation

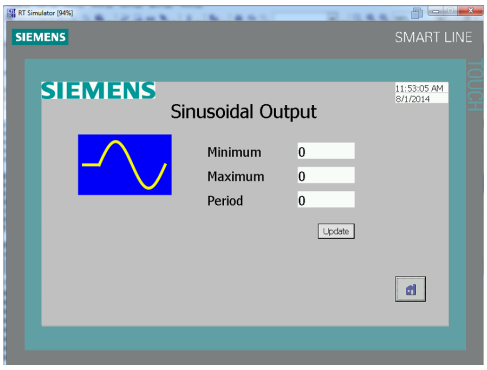


Fig. 14. Input Screen for Sin Signal

Cosine functions in Equation 1 were replaced by Taylor series expansion to the second degree. Model analysis showed that second order Taylor series approximation was good enough to produce accurate results. Figure 13 shows a graphical representation of the original (brown) and approximated (blue and green) math models.

Figure 13 indicates the that the math model can be approximated with excellent accuracy using two pieces of Taylor series expansions about two different points.

D. Algorithm Built-in Redundancies

It should be noted that the implemented algorithms incorporated intrinsic redundancies and therefore sensor inputs could be further minimized. In the current design, 4 resistor gauge meters were used to monitor the eccentricity of the actuation ring, which was mathematically redundant. Theoretically, 3 pieces of independent displacement information were enough to find the geometric position of the ring, and in engineering practices the required number of independent sensor inputs could be further reduced to only 2, with additional knowledge of actuator positions and laws of physics.

However, the existence of built-in redundancies in the algorithms does not imply inferior efficiency of performance. First of all, these redundancies involved in calculation of eccentricity largely eased computation and evened out sensor uncertainties. In theory, the system can withstand the loss of 1 or 2 resistor gauge meters without losing its algorithmic integrity. For simplicity, the fail-safe features mentioned above were not implemented but could be incorporated in future improvements.

VIII. Final Design

A. Electrical and Electronic Part

The PLC collects information from the four displacement sensors and one of the two angle sensors; it then calculates the pushing or pulling force that would minimize the difference between actual average vane angle and the desired angle; signal is sent as voltage to the transducer, in order to control the revolving speed of the actuator. The actuator then push a lever connected to the ring with screw stricture.

The sensors works as resistors with variable resistance, when used to detect angles and displacements. The analogue-digital converting module will give the digital value of the voltage, by which we can determine the displacements. The mathematical relationship between the sensors and the angles enables the CPU of PLC to calculate the angles and determine what to do, including deciding the way the actuator should move, as well as the speed of actuator rotation. The correct reaction is then sent to the transducer to minimize the difference.

As we can see in subsection D-C, there are altogether 6 subroutines in the whole project. The first one (`init`, `SBR7`) initializes the PLC, including fetching the address of the data array, which will be called only once, instead of in every scan cycle, like other subroutines. The second one (`set`, `SBR6`) identifies the type of desired motion in this cycle from the HMI; the typed value will be tested to make sure that it is valid, otherwise the internal parameters will not change. The routine `read_inputs` (`SBR0`) and `cal_phi` (`SBR2`) reads the displacement sensors a_1 to a_4 and angle sensor θ_0, according to which it calculates the eccentricity and rotation of the actuating ring as we described in subsection VII-B. Then `cal_angle` (`SBR3`) will give the angle for every vane. Finally, the PID module, given by the PID wizard from STEP 7, tune the output signal according to the difference between the desired value and present average angles. As analyzed in subsection VII-C, all those commands are expected to finish within one cycle.

Users control the PLC with an HMI panel produced again by Siemens. They are connected with a PPI (Point to Point Interface) cable with connector DE-9 and protocol RS-485 (the protocol itself does not specify a connector type). The HMI touch panel has a screen to start the motion and set parameters for either type of motion, as shown in Figure 14 and Figure 15; touching the input region will activate a virtual keyboard to enter values, and the button "Update" should be pressed before the entered value actually take effect. If user switches between modes, the switching is started by pressing the button "Update", instead of switching the screens. Angles of all the ten vanes are shown in Figure 16; users can view the precise value in the table below the chart, with corresponding time changed by tapping the chart. Due to the hardware limitation, a line chart can only display four groups of values.

B. Mechanical Part

a) Overview: After concept selection, the mechanical design was carried out. Figure 17 shows the overview of the

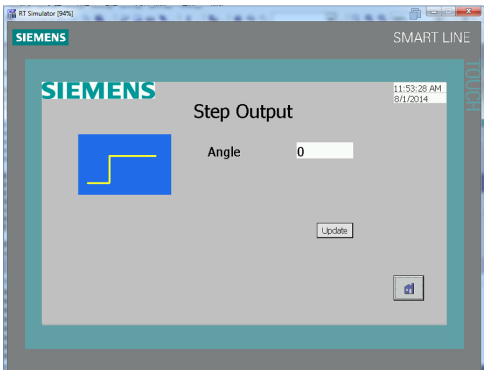


Fig. 15. Input Screen for Step Signal

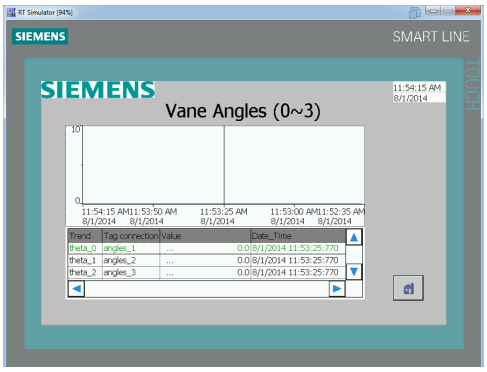


Fig. 16. Output Screen for Angles of Vane 0˜3

mechanical part which involved four main part: Shell, Ring, Transmissions and Support. Each vane connected with ring by linkage, which let the ring can drive the vane rotate from 10° to 90°.

b) Shell: Original shell was recycled to reduce the total cost. Some further modification was done to meet our design. To attach the support on the shell, eight axis are fixed on the shell by fastening screw. Shaft sleeve is used to connect the vane and shell and shaft shoulder was attached on the axis of vane so as to hold the vane in the right position. Figure 18 shows the shell with several hole, and the screw hole is marked by the green circle.

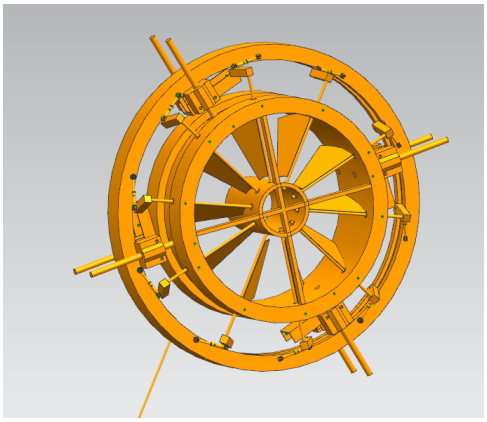


Fig. 17. Overview of Mechanical Part

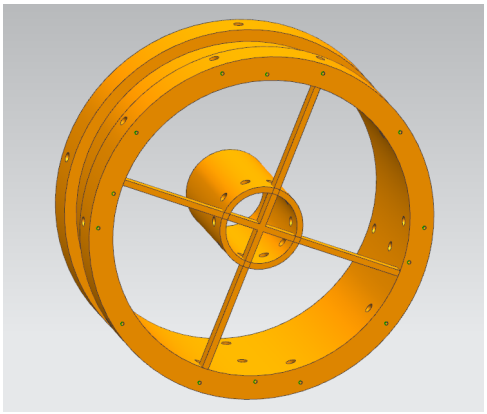


Fig. 18. Design Detail — Shell

Fig. 19. Overview of the Ring

c) Ring: A new aluminum ring with a 0.5 cm, which is in Figure 19, slot is manufactured to instead plastic ring done by pervious team. It has better stiffness and accuracy. The slot can help the roller in support fit the ring perfectly which let the whole system more stable. Also the actuator and vanes are connected on the ring by axis and transmission part which cause the ring become the center pin. So the working accuracy is very important in this part which also increase the cost.

d) Transmissions: An eye-ball linkage and a normal linkage were used in each transmission part which is shown in Figure 20. Two linkage can give the system enough degrees of freedom for our design. The Eye-ball linkage has six degrees of freedom which can simplify the install the transmission part and lower the fiction. Screw is used to connect the normal linkage, eye-ball linkage and ring while the normal linkage is attached on the axis of vane by fastening screw.

e) Support: A new support has been designed to minimize the axial movement while it will not influence the radial and tangential movement. Two linear bearings which are shown in Figure 21 were attached in the support part is to avoid the axial movement while the radial movement works as usual. The roller is matched the slot, which has been mentioned

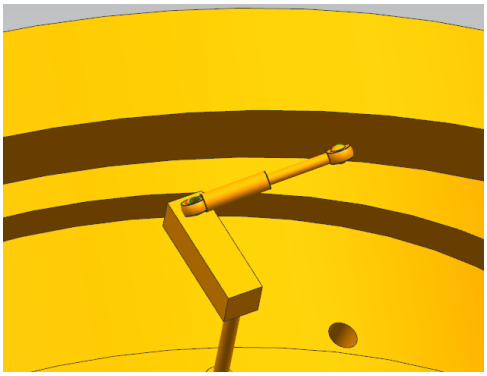


Fig. 20. The linkages in Transmission Part

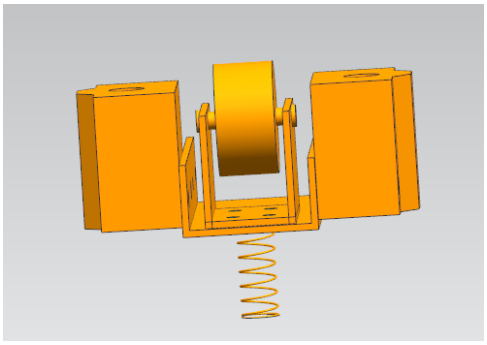


Fig. 21. Overview of the Support Part

in the ring part, perfectly. So the rotation of the ring would not be influenced by the support part. Each support is attached on the shell by two axis and a spring. Axis make sure that the axial movement is avoided and the spring make sure that it support can move up and down so as to let the ring move freely.

IX. Manufacturing Plan

Figure 22 is an overview of the whole IGV Actuation Control System. Please refer to the bills of material (section B) for purchase and manufacturing information of each part.

The manufacture of the system started from the mechanical part. Refer to Figure 23 through Figure 25 for labelling of components.

1) Install the vanes with axes in the Shell. Remember to put in the PTFE axle housing and shaft shoulders first.

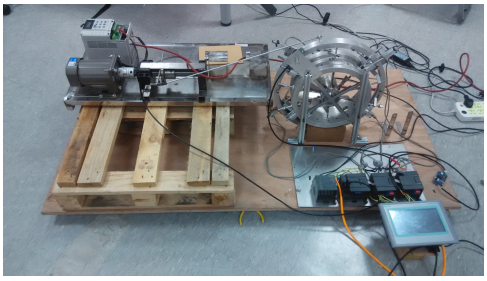


Fig. 22. Manufactured prototype

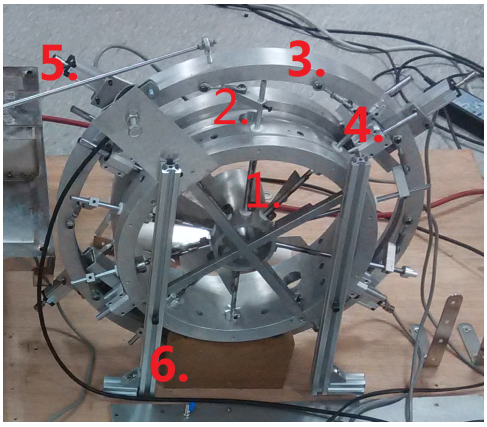


Fig. 23. Prototype component labeling — 1-6

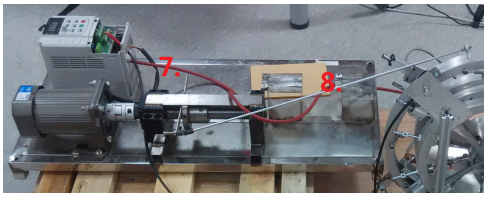


Fig. 24. Prototype component labeling — 7-8

2) Adjust the vanes to proper location according to the CAD sketch. Put on the linkage and screw it.
3) Link the actuation ring with the ten linkages with screws.
4) Install the roller support — install one axis first; Then adjust the roller to proper location and install the second axis.
5) Install four resistance gauge meters and the two angle transducer.
6) Fix the ring and shell one the wooden board using aluminum rods.
7) Put the frequency changer actuation motor on the steel board.
8) Connect the universal joint on the motor. Use a long steel rod to connect the ring and the motor.
9) Fix the PLC module and the power module on the board.
10) Connect all wires and power supplies. Please refer to the appendix of wire connection.

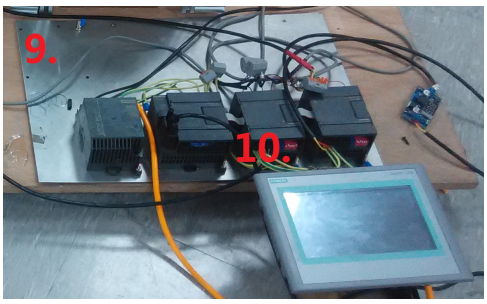


Fig. 25. Prototype component labeling — 9-10

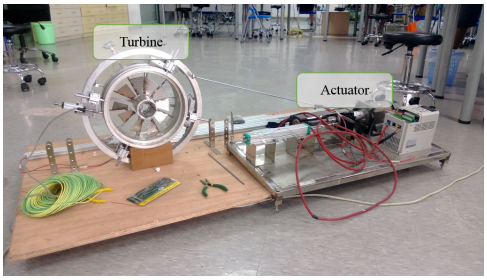


Fig. 26. Setup of the First Test

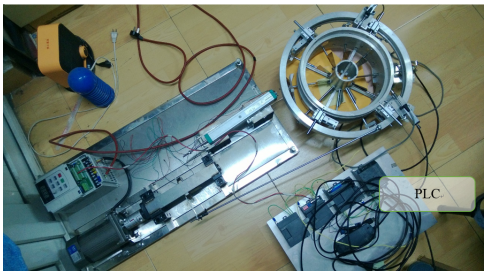


Fig. 27. Setup of the Second Test

X. Test Results

A. Test without PLC or HMI

The purpose of the project is control the angle of the vane, thus to adjust the amount of inlet air. Within the scope of the project, the effect of gas flow is not considered. The experiment was divided into two parts. It is running without HMI and PLC at first. Figure 26 was the setup of the first test. It was a simple test, because the only thing needed to be proved is that the vane can change its angle through the driving force of the actuator. The motor first drove the rod, which was connected to the outer ring. Then the ring can change the vane angle by the linkage. The results showed that the vane can move with excellent smooth. Because there was no appearance of PLC, the motor would not go back once it reached the boundary position of the vane.

PLC was added during the second round of experiment in order to test the computer part. Because this experiment was set in the dorm, the turbine was laid down on the floor. In Figure 27, PLC was placed alongside the actuator and the turbine. Through the help of PLC, a certain range of degree could be set inside. The purpose was then to see whether the actuator would go back as programmed. Table IV was the result of this experiment. The actual vane angle was measured by human. Because it was an early test, a deviation under 10 degree could be accepted.

The final test was a combination of all the components in the project, involving turbine, actuator, PLC, and HMI. The interface was placed near the PLC, which is at the lower right of Figure 28. Now the angle could be input into the

TABLE IV. Result of Test Round Two

Range of the angle (°)	Actual average angle range	Pass (Y) or not (N)
10-80	11-72	Y
10-45	13-51	Y
45-80	41-71	Y

TABLE V. Result of Test Round Two

Range of the angle (°)	Actual average angle range	Pass (Y) or not (N)
10-80	10-76	Y
10-45	10-50	Y
45-80	44-76	Y

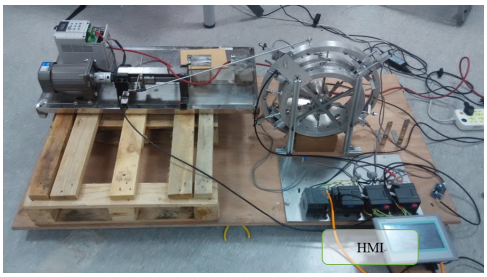


Fig. 28. Setup of the Third Test

PLC directly through the HMI. The same angles as test two were chosen again. This time the deviation target was set as 5 degree.

XI. Engineering Changes Notice

The only change since Design Review 3 is the addition of shaft shoulders, which is the ring in Figure 29. It is attached on the axis of each vane, and next to the inside surface of the shell. This can prevent the vane from falling out. The Engineering Changes Notice is illustrated in Figure 30.

XII. Discussion

In this section, improvements of the current IGV Actuation Control System to the previous design would be introduced; the lack of the design and the recommendation of development would also be discussed.

A. Improvement

A main focus of IGV Actuation Control System was making improvements to the previous one. As what had been mentioned, there existed several parts, which were less than ideal, caused the cut down of precision in the previous design.

f) Aluminum Alloy Ring: Previous design implemented a plastic ring as the actuator in the design, which was cheap and easy to manufacture. However, the elastic and soft property of plastic made the system unstable: deformation of ring caused the calculation of vane angles differed from the ideal values; the difference between each vanes angle became

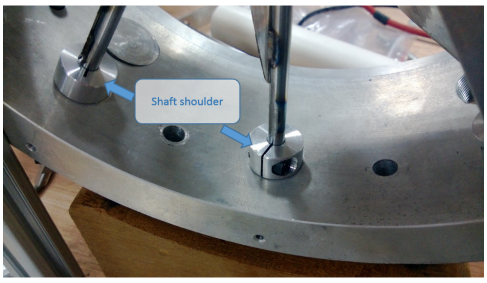


Fig. 29. Shaft shoulder and its position

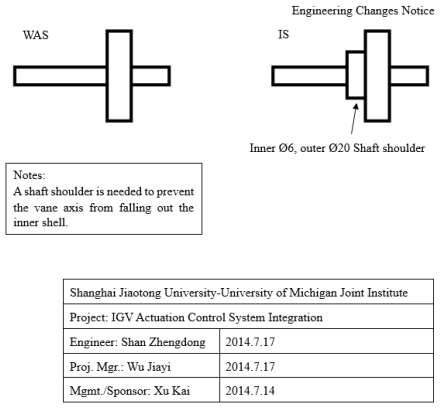


Fig. 30. Engineering changes notice

inevitable; and since its contact with the support (between the ring and the shell) was not fixed, the axial movement would occur, and enlarged the system error further. To get rid of those problems, the aluminum alloy ring was implemented. The aluminum alloy ring had high stiffness and thus avoided the deformation; the groove on the ring also prevented the support from moving in radial direction (this problem would be introduced in the next part). Although the cost had been raised to 1000 RMB on the ring, this change had been assessed as effective.

g) Roller Support: The support structure (between the ring and the shell) of the previous design was called a spring structure (Team+4-Final+Report.pdf, Fall 2013). It was consisted of a steel ball to reduce the friction when the ring moved, a spring to lift the ball, and a sleeve to maintain the spring and the ball. Since the ball could roll in every direction, movement of ring in radial direction became obvious. The radial movement problem was serious since its the most essential error in the system. In comparison, the roller support did not have this problemthe pulley on the support could only roll in one direction. In addition, groove on the ring provided a rail to the roller, which prevented the radial movement further. It was a complex design compared to the spring structure, but it solved the problem.

h) Resistor Gauge Meter: To calculate the eccentricity, the displacement sensor should be used in the IGV system, and the previous team chose drag wire displacement senor. This sensor had a range of 1000 mm, and a precision of 0.3-0.1%FS. The resistance gauge meter had a range of 25 mm, and a precision of 0.1%FS. Since the eccentricity of the ring was less that 20mm, the resistance gauge meter gave a enough measuring range and reduced the error for 40 times at least.

i) External Measurement of Vane Angle: The external measurement meant the directly measurement of vanes angles, which could be compared with the expected value on HMI. To realize the measurement, another angle transducer was installed on the system. It could measure the angle of one vane once, but its installation frame could be easily installed and uninstalled so that the measurement of multiple vane angles was possible.

j) Feedback Control: The feedback control enabled the vane to adjust its angle via the data from angle transducer. For instance, when a given a target angle to the IGV system, the vane would not reach this target once and for all. This was caused by the huge friction and of large driving force of motor. Knowing this deviation (according to the data provided by the angle transducer), the system would adjust gently until reaching the target angle. Feedback control offered a more reliable way for vane angle control.

B. Recommendation

In order to improve the previous design, significant change has been made in the IGV Actuation Control System design. Meanwhile, those changes brought new problems. Although the current team managed to solve most of the problems, there were still some left due to the limitation in time and resource.

k) Machining Precision: According to the design guideline and the error evaluation, an ideal machining precision should be less than ± 0.1mm. Under this condition, the deviation among vanes would be less than 1.8° (which met the customer requirement). However, this precision was hard to be realized in all parts with the limited budget we had. For instance, multiple threaded holes had been drilled on the ring and the shell. They were used to fix the linkages, vanes, and the roller support. With the help of protractor, the holes along axial-direction could be drilled properly, but holes along radial-direction were messed (since no appropriate tools could be used for help). In other situations when manufacturing work required the operation by hand directly, visible errors were inevitable. Manufacturing seems to be the predominant source of uncertainties in this project.

l) PLC Speed: Siemens S7-200CN PLC Module was a stable programmable logic controller commonly implemented in industrial electric equipment. The flaw of it was its speed. The calculation capacity of the module was too small. It would take 1 millisecond to calculate a simple sine function (it was a lone time for a computer). Due to the limitation in design budget, a more powerful PLC module was not affordable. In this design of IGV Actuation Control System, Taylor series expansion was implemented to approximate the sine function to reduce the calculation time. If a more powerful PLC module could be used, the system could provide a faster response time.

XIII. Conclusion

IGV Actuation Control System design is a daunting and highly technical task that requires comprehensive understanding of the problem and lots of novelty as well as insights during the design and manufacturing process. Compared to previous teams working in the project series, a lot of improvements and design changes were made to correct errors and improve the performance. Major improvements include: a redesign of the support mechanism, a refinement of the transmission mechanism, a complete analysis of the math model and a newly proposed sensing/algorithm combination, and complete human-machine interaction as well as software implementations. Despite limitations and room for future improvements, the test results showed our success in recognizing major customer requirements. The project proved the concept and feasibility of the proposed solution to the stated problem.

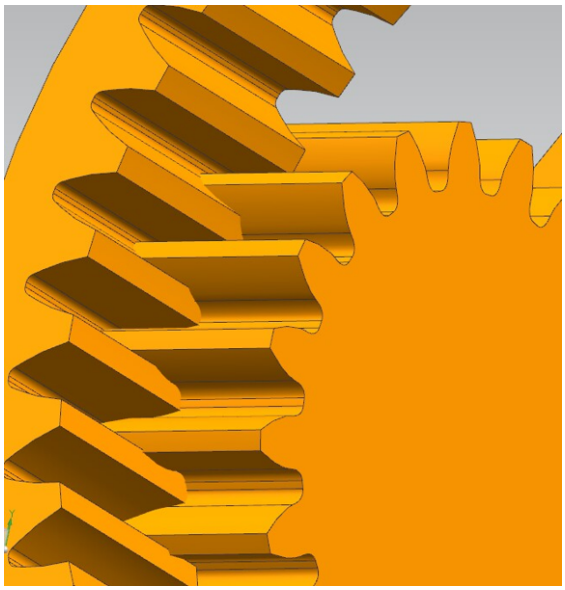


Fig. 31. Generated concept: gear-based transmission

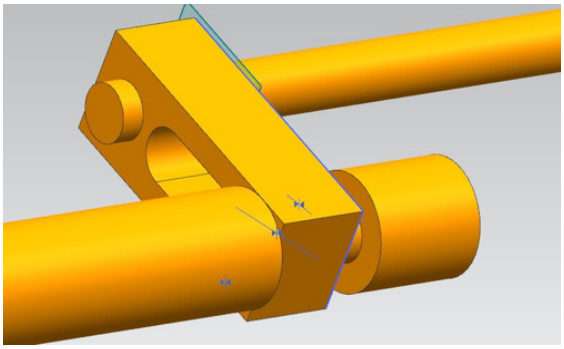


Fig. 32. Generated concept: slider-based transmission

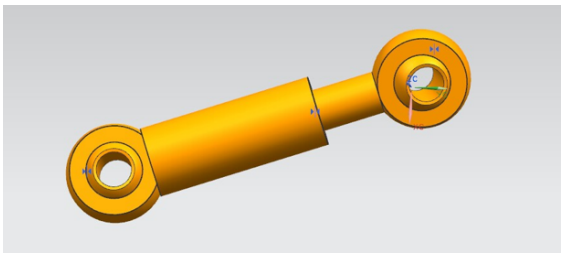


Fig. 33. Generated concept: eye-bolt-based transmission

ACKNOWLEDGMENT

The authors would like to thank Mr. Pan Wenyong and Mr. Lau Ow from Siemens for their sponsorship and technical support during the project. The authors would also like to thank all three previous teams working in the project series for their exploration; contribution and insights from previous work eased our design and manufacturing process and shielded us from certain mistakes. Finally, the authors would like to show their greatest respect and appreciation to Professor Xu Kai from UM-SJTU Joint Institute for his support, insights, and suggestions throughout the project.

REFERENCES

[1] K. Fuji and K. Toyama. *Inlet Guide Vane Control Device of Gas Turbine*. Pat. US 7,422,414 B2. 2008.

[2] J-W Kim and S.W. Kim. *Design of Incremental Fuzzy PI Controllers for a Gas-Turbine Plant*. In: IEEE/ASME Transactions on Mechatronics. Vol. 8. 2003, pp. 410414.

[3] T.D. Mahoney et al. *Gas turbine engine electromechanical variable inlet guide vane actuation system*. Pat. US 7,096,657 B2. Aug. 29, 2006.

[4] G. Mannarino. *Control System for Positioning Compressor Inlet Guide Vanes*. Pat. US 2004/0055310. 2004.

[5] G.S Stoner and P.M. Schoonmaker. *Inlet Guide Vane Assembly*. Pat. US 6,039,534 A. 2000.

[6] S. Tanaka. *Gas Compressor Control Device and Gas Turbine Plant Control Mechanism*. Pat. US 6,907,722 B2. 2005.

[7] E. Tremaine and A.B. Newland. *Variable Inlet Guide Vane Mechanism*. Pat. US 4,890,977 A. 1990.

[8] *SIEMENS S7-200 system manual*, http://www.automation.siemens.com/doconweb/pdf/SINUMERIK_SINAMICS_04_2010_E/S7200SH.pdf?p=1.

[9] *x86 asm reference*, http://x86.renejeschke.de/html/file_module_x86_id_114.html

APPENDIX A
CONCEPT DETAILS AND COMPARISONS

A. Transmission

Generated concepts for transmission included a gear-based solution (Figure 31), a slider-based solution (Figure 32), and a eye-bolt-based solution (Figure 33). Figure 34 illustrates how eye bolt linkages work in the context of the stated problem.

Each concept has its own advantages and disadvantages. The gear-based concept yields better precision but also incurs high manufacturing and maintenance costs. The slider-based concept requires the least amount of linkages per mechanism, but is relatively less flexible and the slider structure subjects to substantial wear and tear during operation. The eye-bolt-based one provides sufficient degrees of freedom to ensure its

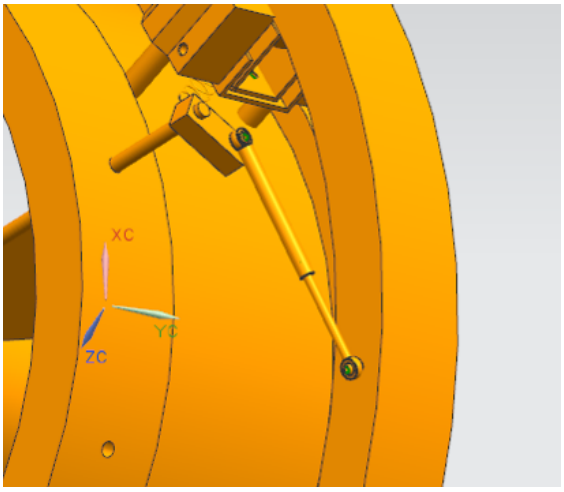


Fig. 34. Functional illustration of eye bolt linkages

TABLE VI. SCORING MATRIX FOR SELECTING CONCEPTS OF TRANSMISSION

	Weight	Gear	Slider	Eye bolt
Accuracy/Precision	*9*	5	3	3
Service Life	*10*	2	3	4
Cost-effectiveness	*4*	1	5	4
Flexibility	*7*	1	2	5
Total (weighted)		76	91	118

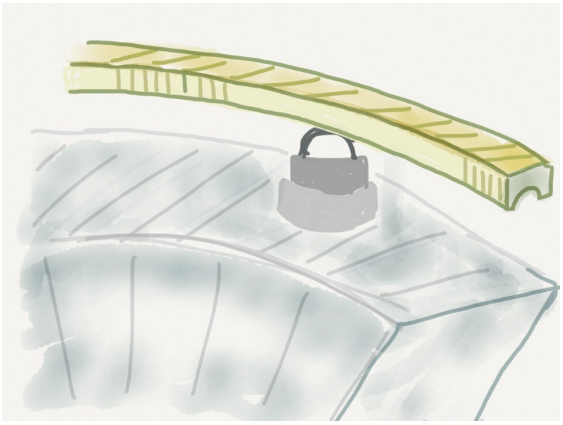


Fig. 35. Generated concept: the original support mechanism

Fig. 36. Generated concept: draw-wire displacement sensor

Fig. 37. Generated concept: resistor gauge meter

flexibility, but could compromise the accuracy and precision of vane angle control. The scoring matrix shown in Table VI was used to choose the winner concept.

B. Support

Apart from the selected concept shown in Figure 8 in section VI, the original concept was carefully examined to identify room for improvement. The concept of the original support mechanism is illustrated in Figure 35. The main problem with the original design was that it did not provide sufficient constriants along the axial direction to prevent the ring's undesired movement. The socring matrix chosing the winning concept is shown in Table VII.

C. Sensing

Three sensing concepts were generated and compared against each other. They include:the drag-wire displacement sensor, the resistor gauge meter, the force sensitive resistor, and the piezometer (shown in Figure 36, Figure 37, Figure 38, Figure 39, respectively).

The idea of indirectly measuring displacements from pressure sensors was discarded because of the lack of accuracy. Force sensitive resistors have very poor linearity as well as repeatability, and piezometers are not designed to measure pressure in quasi-static conditions anyways. It was concluded during the design decision making process that displacements of the acutation ring must be directly measured using displacement sensors.

Resistor gauge meter was chosen over the drag wire displacement sensor because it has a higher precision. The range of resistance gauge meter is 25 mm; while the range of drag wire displacement sensor is 1000 mm. Having similar full scale inlinearity and repeatability, resistance gauge meter has much less error. (Here F-sensitive R means *force sensitive resistor*, and RG-sensor means *resistor gauge meter*.) The detailed scoring matrix was shown in Table VIII.

TABLE VII. SCORING MATRIX FOR SELECTING CONCEPTS OF SUPPORT

	Weight	Original	New
Accuracy/precision	*9*	1	5
Service Life	*10*	2	5
Cost	*4*	5	2
Friction	*7*	3	5
Total (weighted)		70	138

APPENDIX B
BILLS OF MATERIALS

Table IX shows the Bills of Materials of the project.

TABLE VIII. SCORING MATRIX FOR SELECTING CONCEPTS OF SENSING

	Weight	Drag Wire	F-Sensitive R	Piezometer	RG-Sensor
Accuracy/Precision	*9*	2	4	3	5
Service Life	*10*	3	4	4	4
Cost-Effectiveness	*4*	2	4	4	4
Efficient Use of Sensors	*3*	1	3	2	4
Total (weighted)		59	101	89	123

TABLE IX. BILLS OF MATERIALS

Item	Quantity	Source	Catalog	Number	Cost (RMB, Total)	Contact	Notes
Aluminum Ring	1	ECUST		980	ECUST		
Aluminum Shell	1	SJTU		1000	SJTU	Legacy	
ϕ6*100mm Steel axis	10	Taobao		40	Taobao.com		
ϕ8*100mm Steel axis	8	Taobao		18	Taobao.com		
ϕ1.2*25mm Spring	4	Taobao		3.2	Taobao.com		
1" Pulley	4	Taobao		24.8	Taobao.com		
C-shaped Aluminum Alloy	4	Taobao		175.3	Taobao.com		
KTR-25MM Resistance Gauge Displacement Sensors	4	Taobao		760	Taobao.com		
M2500 Angle Transducer	2	Taobao		350	Taobao.com		
Siemens Smart 700IE HMI	1	Siemens		1000	Siemens		
Aluminum Linkage	10	ECUST		100	ECUST		
Eyeball linkage	10	Taobao		37	Taobao.com		
S7-200CN PLC Module	1	SJTU		920	SJTU	Legacy	
SITOP PS207 Power Module	1	SJTU		320	SJTU	Legacy	
A.Q.L Motor 220V 200W	1	SJTU	6IK200K-SF	700	SJTU		
Variable Frequency Drive	1	SJTU	Rilipu RLP-2R2G-S2-485-Z	750	SJTU	Legacy	
M3 screws	20	Taobao		0.2	Taobao.com		
M4 screws	10	Taobao		0.1	Taobao.com		
M5 screws+ screw nut	4	Taobao		0.1	Taobao.com		
6 PTFE axle housing	20	Taobao		53	Taobao.com		
6 shaft shoulder	10	Taobao		150	Taobao.com		
Linear bearing	8	Taobao		73.2	Taobao.com		
6 Universal joint	1	taobao		12.5	Taobao.com		

Fig. 38. Generated concept: force sensitive resistor

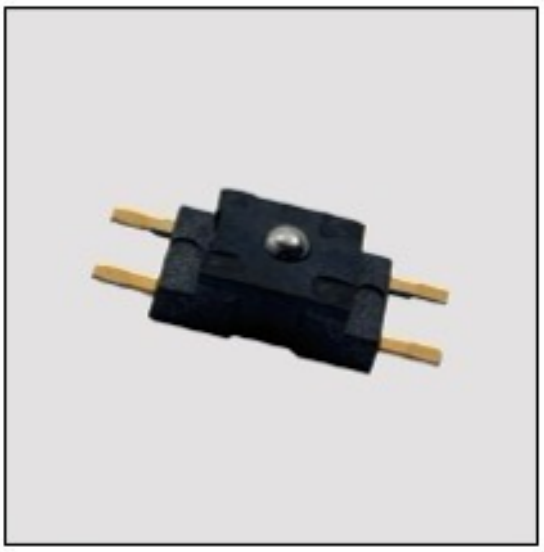
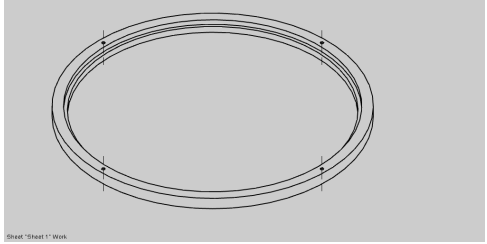
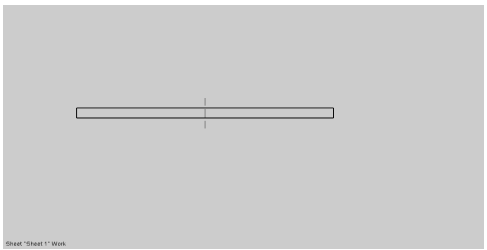
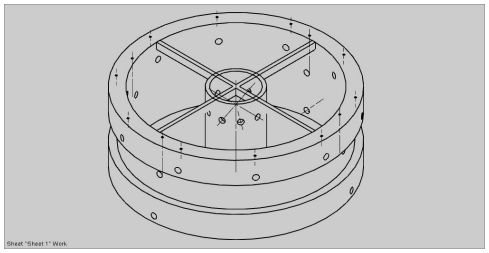


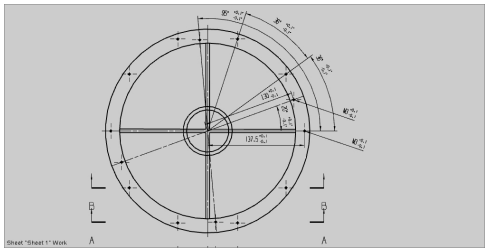
Fig. 39. Generated concept: piezometer

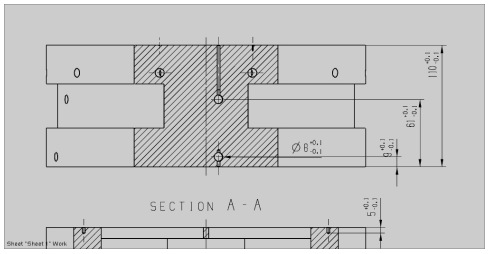
APPENDIX C
DRAFT FOR MANUFACTURING

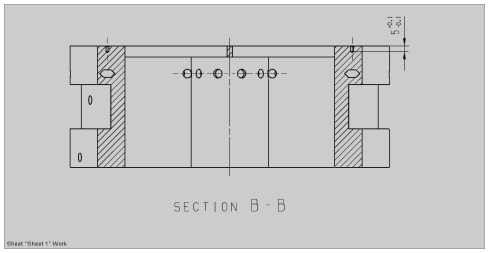


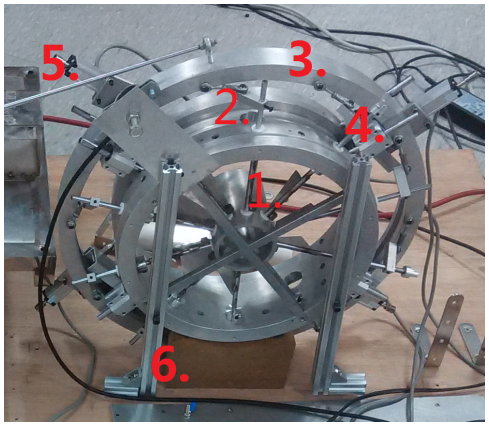












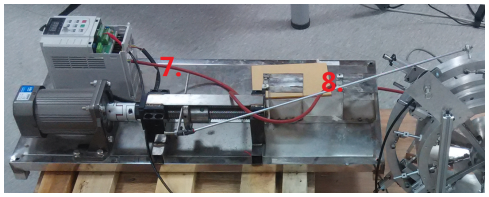
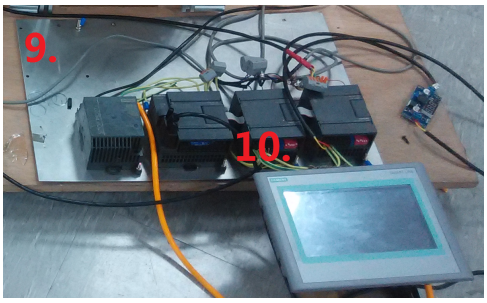
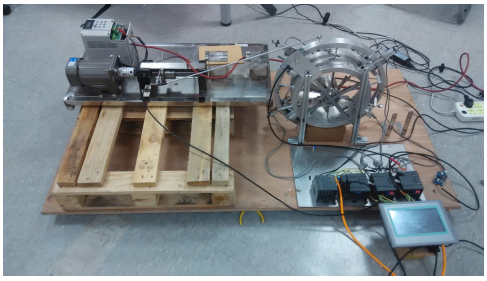


TABLE X. DIMENSIONAL QUANTITIES IN MATH MODEL CALCULATION

Quantity	Value (mm)
R	195.80
r	25.00
x_0	-42.25
c	40.00
L	188.07





APPENDIX D
IMPLEMENTATION OF CONTROLLER SOFTWARE

Software that runs on the SIEMENS S7-200 Programmable Logic Controller (PLC) was developed and tested using the SIMATIC STEP-7 MicroWin v4.0 SP9 development toolkit under 64-bit Windows 7 environment.

A. *Math Model Calculation*

Dimensional quantities involved in math model calculation is shown in Table X. These quantities were used to reduce and simplify the expression derived from Taylor series expansion. See section VII for descriptions of dimensional quantities.

The equivalent C code of the PLC program responsible for math model calculation is shown below.

```
/*
 *  PLC Program in C
 *  > The PID control loop is implemented using
     the built-in PID instruction
 *     and is not reflected in this program.
 *  > The 'float' datatype corresponds to 'real'
     type in S7-200 STL language.
 *  > Updated by Yihe Huang on 07/29/2014 to
     reflect recent changes regarding
 *     the use of external libraries.
 */

// Pre-computed constants
// tan(75d)=3.732, sin(75d)=0.9659

// deg  cosine  sine
//   5d .9962   .08716
//  41d .7547   .6560
```

```
//  77d .2250   .9744
// 113d -.3907  .9205
// 149d -.8572  .5150
// 185d -.9962  -.08716
// 221d -.7547  -.6560
// 257d -.2250  -.9744
// 293d .3907   -.9205
// 329d .8572   -.5150

// Sensor offsets (to be adjusted during
    calibration)
const float disp_offsets[4];
const float angle_offset;

// Rotation matrix coefficients
const float sine_table[10] = {0.08716, 0.6560,
    0.9744, 0.9205, 0.5150, -0.08716, -0.6560,
    -0.9744, -0.9205, -0.5150};
const float cosine_table[10] = {0.9962, 0.7547,
    0.2250, -0.3907, -0.8572, -0.9962, -0.7547,
    -0.2250, 0.3907, 0.8572};

// Sensor input registers
uint16_t d_sensors[4];
uint16_t a_sensor;
uint16_t full_scale_raw;

// Global variables
float full_scale_f;
float angles[10];
float n_dy, n_dz;
float phi, phi_2;
float average_angle = 0; // Control feedback

void main() {
    // Scan cycle starts
    // angles[0] contains measured theta
    read_inputs();
    calculate_phi();
    for(int i = 1; i < 9; ++i) {
        calculate_theta(i);
    }
    // Scan cycle ends
}

void read_inputs() {
    // Read, convert, and scale analog inputs
    // Scalings are done using SIMATIC Scale
        Library v1.2
    float temp;
    full_scale_f = (float)full_scale_raw;
    n_dy = (float)d_sensors[0]/full_scale_f*25.0
        - disp_offsets[0];
    n_dz = (float)d_sensors[2]/full_scale_f*25.0
        - disp_offsets[2];
    n_dy = (n_dz - n_dy)/2;
    temp = (float)d_sensors[1]/full_scale_f*25.0
        - disp_offsets[1];
    n_dz = (float)d_sensors[3]/full_scale_f*25.0
        - disp_offsets[3];
    n_dz = (n_dz - temp)/1.932 + l_dy/3.732;
    angles[0] = (float)a_sensor/full_scale_f
        *355.0 - angle_offset;

    // update average
    average_angle = average_angle * 0.9 + angles
        [0] * 0.1;
}

void calculate_phi() {
    // COMPLEXITY: 27 mult, 2 div, 9.5 add, 11.5
        sub
    // TIMING(ms): 7.032 max, 3.368 typ
    float l_dy, l_dz; // Local displacements (
        eccentricity)
    float theta_2;
    float a, b, c; // Quadratic equation
        coefficients, see report for more
        information of
                    // approximation techniques

    // Apply rotatation matrix
    l_dy = 0.9962*n_dy + 0.08716*n_dz;
    l_dz = -0.08716*n_dy + 0.9962*n_dz;
    theta_2 = angles[0] * angles[0];

    // Apply piece-wise 2-nd order Taylor's
        series expansion approximation
    if(angles[0] < 4) {
        a = -176.223*theta_2 - 23.496*l_dy +
            36997.917 + 410.20*angles[0] -
            194.433*l_dz;
        b = 610.907*l_dz - 109037.696 + 5565.717*
            angles[0] - 320.335*l_dy - 2398.594*
            theta_2;
        c = 87623.3 + l_dz*l_dz + l_dy*l_dy +
            5046.043470*theta_2 + 14.950*l_dy*
            theta_2 - 464.4438236*l_dz
            - 15246.68803*angles[0] - 34.750*l_dy
                *angles[0] + 524.6231176*l_dy;
    } else {
        a = 290.7683460*theta_2 + 42445.66269 -
            2741.250400*angles[0] - 194.4329748*
            l_dz - 23.4964320*l_dy;
        b = -35102.25650 + 610.9072320*l_dz -
            320.3346896*l_dy - 37251.63490*angles
            [0] + 3950.337630*theta_2;
        c = l_dz*l_dz + l_dy*l_dy - 6750.544760*
            theta_2 - 54978.00727 + 65801.36756*
            angles[0]
            + 232.600*l_dy*angles[0] - 24.675*
                l_dy*theta_2 - 464.4438236*l_dz +
                 62.9731176*l_dy;
    }

    phi = (-b - sqrt(b*b - 4*a*c)) / 2 / a;
    phi_2 = phi * phi;
}

void calculate_angle(int i) {
    // COMPLEXITY: 29 mult, 2 div, 12 add, 11 sub
    // TIMING(ms): 7.562 max, 3.634 typ
    float l_dy, l_dz;
    float a, b, c;

    // Apply rotation matrix (clockwise)
    l_dy = cosine_table[i]*n_dy + sine_table[i]*
        n_dz;
    l_dz = -sine_table[i]*n_dy + cosine_table[i]*
        n_dz;

    // Apply piece-wise 2-nd order Taylor's
        series expansion approximation
    if(angles[0] < 4) {
        a = 5046.043470 - 2398.594100*phi +
            14.950*l_dy - 176.2232400*phi_2;
        b = -34.750*l_dy + 5565.717330*phi -
            15246.68803 + 410.2085420*phi_2;
        c = 610.9072320*l_dz*phi - 194.4329748*
            l_dz*phi_2 - 23.4964320*l_dy*phi_2 -
            320.3346896*l_dy*phi + 87623.30094
            + l_dz*l_dz + l_dy*l_dy +
                36997.91703*phi_2 - 109037.6959*
                phi + 524.6231176*l_dy -
                464.4438236*l_dz;
    } else {
        a = -24.675*l_dy - 6750.544760 +
            3950.337630*phi + 290.7683460*phi_2;
        b = 232.600*l_dy + 65801.36756 -
            37251.63490*phi - 2741.250400*phi_2;
```

TABLE XI. SELECTED CONTROLLER PARAMETERS (PI CONTROLLER)

Parameter	Description	Value
K_P	Loop gain	-2.8
T_I	Integral time (sec)	0.7
T_D	Differential time (sec)	0 (Disabled)
T_S	Sample time (sec)	0.1

```
        c = -194.4329748*l_dz*phi_2 - 23.4964320*
            l_dy*phi_2 - 320.3346896*l_dy*phi -
            54978.00727 + l_dz*l_dz + l_dy*l_dy
            + 62.9731176*l_dy + 42445.66269*phi_2
                - 35102.25650*phi - 464.4438236*
                l_dz + 610.9072320*l_dz*phi;
    }

    angles[i] = (-b + sqrt(b*b - 4*a*c)) / 2 / a;

    // Update average
    average_angle = average_angle * 0.9 + angles[
        i] * 0.1;
}
```

B. Control Algorithm

Standard PI control was implemented by the native PID instruction provided by the PLC. The PID configuration Wizard within the Step-7 development toolkit was used to generate and tune PI controller gain values. Table XI summarizes the chosen PID controller parameters.

C. Source Code

The actual code that runs on the PLC was implemented using the statement list (STL) language. All newly developed code (including the interface with the HMI and initialization of control parameters) in STL is shown below.

```
ORGANIZATION_BLOCK MAIN:OB1
TITLE=control program for vane
VAR
i:WORD; // iteration variable
END_VAR
BEGIN
Network 1 // Initialization
// Initilize pointers in the symbol table. Run on
     the first scan only.
LD     SM0.1
CALL   SBR7

Network 2 // Network Title
// read_inputs ();
// PID initialization
// calculate_phi ();
LD     SM0.0
CALL   SBR6
CALL   SBR0
CALL   SBR5, VD156, 6.19592, 0.0, 32000, 0, VW200
CALL   SBR4, AIW8, VD204, AQW0

CALL   SBR2

Network 3
// For loop
LD     SM0.0
FOR    LW0, 1, 9
ITD    LW0, AC1
CALL   SBR3, AC1
Network 4
// end For
NEXT
END_ORGANIZATION_BLOCK
SUBROUTINE_BLOCK read_inputs:SBR0
TITLE=void read_inputs ();
VAR
temp:REAL;
END_VAR
BEGIN
Network 1
// n_dy = (float)d_sensors[0]/full_scale_f -
    disp_offsets[0];
LD     SM0.0
MOVR   VD164, AC1
AENO
+R     *VD176, AC1
AENO
CALL   SBR1, *VD184, VW180, 0, AC1, *VD176, VD140
Network 2
// n_dz = (float)d_sensors[2]/full_scale_f -
    disp_offsets[2];
LD     SM0.0
MOVD   VD184, AC1
AENO
+D     +4, AC1
AENO
MOVD   VD176, AC2
AENO
+D     +8, AC2
AENO
MOVR   VD164, AC3
AENO
+R     *AC2, AC3
AENO
CALL   SBR1, *AC1, VW180, 0, AC3, *AC2, VD144
Network 3
// n_dy = (n_dz - n_dy)/2;
LD     SM0.0
MOVR   VD144, AC1
AENO
-R     VD140, AC1
AENO
MOVR   AC1, VD140
/R     2.0, VD140
Network 4
// temp = (float)d_sensors[1]/full_scale_f -
    disp_offsets[1];
LD     SM0.0
MOVD   VD184, AC1
AENO
+D     +2, AC1
AENO
MOVD   VD176, AC2
AENO
+D     +4, AC2
AENO
MOVR   VD164, AC3
AENO
+R     *AC2, AC3
AENO
CALL   SBR1, *AC1, VW180, 0, AC3, *AC2, LD0

Network 5
// n_dz = (float)d_sensors[3]/full_scale_f -
    disp_offsets[3];
LD     SM0.0
MOVD   VD184, AC1
AENO
+D     +6, AC1
AENO
MOVD   VD176, AC2
AENO
+D     +12, AC2
AENO
MOVR   VD164, AC3
AENO
```

```
+R     *AC2, AC3
AENO
CALL   SBR1, *AC1, VW180, 0, AC3, *AC2, VD144
Network 6
// n_dz = (n_dz - temp)/1.932 + n_dy/3.732;
LD     SM0.0
MOVR   VD144, AC1
AENO
-R     LD0, AC1
AENO
/R     1.932, AC1
AENO
MOVR   VD140, AC2
AENO
/R     3.732, AC2
AENO
MOVR   AC1, VD144
+R     AC2, VD144
Network 7
// angles[0] = (float)a_sensor/full_scale_f -
    angle_offset;
LD     SM0.0
MOVR   VD96, AC1
AENO
+R     VD188, AC1
AENO
CALL   SBR1, AIW8, VW180, 0, AC1, VD96, *VD160
Network 8
// average_angle = average_angle * 0.9 + angles
    [0] * 0.1;
LD     SM0.0
MOVR   VD156, AC1
AENO
*R     0.9, AC1
AENO
MOVR   *VD160, AC2
AENO
*R     0.1, AC2
AENO
MOVR   AC1, VD156
+R     AC2, VD156
END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK S_ITR:SBR1
TITLE=LIBARY: Scale V1.2 (bipolar scaling)
// ============================
//
// LIABILITY
// Siemens AG does not accept liability of any
    kind for damages arising from the use of this
     application, except where it is obliged to
    by law, in cases such as damage to items used
     for personal purposes, personal injury,
    willful damage or gross negligence.
//
// WARRANTY
// The program examples given are specific
    solutions to complex tasks which were worked
    on by Customer Support. We must also point
    out that it is not possible in the current
    state of the technology to exclude all errors
     in software programs under all conditions of
     use. The program examples were prepared
    according to the best of our knowledge.
    However, we cannot accept any liability
    beyond the standard guarantee for Class C
    software in accordance with our General Terms
     of Sale for Software Products for Automation
     and Drive Technology". The program examples
    can be purchased on the Internet as single
    licenses. They may not be transferred to a
    third party.
//
// PASSWORD
// The password of the library is "1234"
//
```

```
// S_ITR
// Scale Integer to Real
//
//  The formula is as follows:
//  Ov = [(OSH - OSL) * (Iv - ISL) / (ISH - ISL)]
     + OSL
//  with ISL <= Iv <= ISH
//  and OSL <= Ov <= OSH
//
//  Ov  =       output value (REAL)
//  Iv  =       input value (INT)
//  OSH =       high limit of the scale for the
    output value (REAL)
//  OSL =       low limit of the scale for the
    output value (REAL)
//  ISH =       high limit of the scale for the
    input value (INT)
//  ISL =       low limit of the scale for the
    input value (INT)
//
//
//
//
// BIBLIOTHEK: Scale V1.2 (bipolare Skalierung)
// ====================================
//
// HAFTUNGSAUSSCHLUSS
// Bei diesem Programmbaustein handelt es sich um
     FREEWARE.
// Jedem Benutzer steht es frei, dieses Programm
    UNENTGELTLICH zu nutzen, zu kopieren und
    weiterzugeben.
// Die Autoren und Rechtsinhaber dieses Programms
     schlieen jegliche Haftung f r die
    Funktionstchtigkeit oder Kompatibilitt
    dieser Software aus.
// Die Benutzung erfolgt auf eigene Gefahr.
// Da diese Software kostenlos ist, entfllt
    jegliche Gewhrleistung, Anspruch auf
    Fehlerkorrektur und Hotlinesupport.
//
// PASSWORT
// Das Passwort der Bibliothek ist "1234"
//
// S_ITR
// Ganze Zahl in Realzahl skalieren
//
//  Die Formel lautet wie folgt:
//  Ov = [(OSH - OSL) * (Iv - ISL) / (ISH - ISL)]
     + OSL
//  mit ISL <= Iv <= ISH
//  und OSL <= Ov <= OSH
//
//  Ov  =       Ausgangswert (REAL)
//  Iv  =       Eingangswert (INT)
//  OSH =       oberer Grenzwert der Skala f r
    den Ausgangswert (REAL)
//  OSL =       unterer Grenzwert der Skala f r
    den Ausgangswert (REAL)
//  ISH =       oberer Grenzwert der Skala f r
    den Eingangswert (INT)
//  ISL =       unterer Grenzwert der Skala f r
    den Eingangswert (INT)
VAR_INPUT
Input:INT;
ISH:INT;
ISL:INT;
OSH:REAL;
OSL:REAL;
END_VAR
VAR_OUTPUT
Output:REAL;
END_VAR
VAR
Input_DI:DINT;
```

```
ISL_DI:DINT;
ISH_DI:DINT;
delta_R:REAL;
delta_max:REAL;
END_VAR
BEGIN
Network 1 // This POU is password-protected
    against editing and viewing.

END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK cal_phi:SBR2
TITLE=void calculate_phi ();
VAR
l_dy:REAL;
l_dz:REAL;
theta_2:REAL;
aa:REAL;
bb:REAL;
cc:REAL;
END_VAR
BEGIN
Network 1 // Network Title
// l_dy = 0.9962*n_dy + 0.08716*n_dz;
LD      SM0.0
MOVR    0.9962, LD0
*R      VD140, LD0

MOVR    0.08716, AC1
*R      VD144, AC1
+R      AC1, LD0

Network 2
// l_dz = -0.08716*n_dy + 0.9962*n_dz;
LD      SM0.0
MOVR    -0.08716, LD4
*R      VD140, LD4

MOVR    0.9962, AC1
*R      VD144, AC1
+R      AC1, LD4

Network 3
// theta_2 = angles[0] * angles[0];
LD      SM0.0
MOVR    VD100, LD8
*R      VD100, LD8

Network 4
// if angles[0] < 4
LDR<    VD100, 4.0
=       S0.0
Network 5
// then
LSCR    S0.0
Network 6
// a = -176.223*theta_2 - 23.496*l_dy + 36997.917
     + 410.20*angles[0] - 194.433*l_dz;
LD      SM0.0
MOVR    -176.233, LD12
*R      LD8, LD12

MOVR    -23.496, AC1
*R      LD0, AC1
+R      AC1, LD12

+R      36997.92, LD12

MOVR    410.2, AC1
*R      VD100, AC1
+R      AC1, LD12

MOVR    -194.433, AC1
*R      LD4, AC1
+R      AC1, LD12
```

```
Network 7
// b = -109037.6959 + 610.9072320*l_dz -
    320.3346896*l_dy + 5565.717330*angles[0] -
    2398.594100*theta_2;
LD      SM0.0
MOVR    610.9072, LD16
*R      LD4, LD16

+R      -109037.7, LD16

MOVR    LD0, AC1
*R      -320.3347, AC1
+R      AC1, LD16

MOVR    5565.717, AC1
*R      VD100, AC1
+R      AC1, LD16

MOVR    -2398.594, AC1
*R      LD8, AC1
+R      AC1, LD16

Network 8
// c = 87623.3 + l_dz*l_dz + l_dy*l_dy +
    5046.043470*theta_2 + 14.950*l_dy*theta_2 -
    464.4438236*l_dz
// - 15246.68803*angles[0] - 34.750*l_dy*angles
    [0] + 524.6231176*l_dy;
LD      SM0.0
MOVR    LD4, LD20
*R      LD4, LD20

MOVR    LD0, AC1
*R      AC1, AC1
+R      AC1, LD20

+R      87623.3, LD20

MOVR    5046.043, AC1
*R      LD8, AC1
+R      AC1, LD20

MOVR    14.95, AC1
*R      LD0, AC1
*R      LD8, AC1
+R      AC1, LD20

//- 464.4438236*l_dz
MOVR    -464.4438, AC1
*R      LD4, AC1
+R      AC1, LD20

//- 15246.68803*angles[0]
MOVR    -15246.69, AC1
*R      VD100, AC1
+R      AC1, LD20

//- 34.750*l_dy*angles[0]
MOVR    -34.75, AC1
*R      LD0, AC1
*R      VD100, AC1
+R      AC1, LD20

//+ 524.6231176*l_dy
MOVR    524.6231, AC1
*R      LD0, AC1
+R      AC1, LD20

Network 9
// end Then;
SCRE
Network 10
// reverse if_phi
LDN     S0.0
=       S0.0
```

```
Network 11
// begin Else
LSCR    S0.0
Network 12
// a = 290.7683460*theta_2 + 42445.66269 -
    2741.250400*angles[0] - 194.4329748*l_dz -
    23.4964320*l_dy;
LD      SM0.0
MOVR    290.7683, LD12
*R      LD8, LD12

+R      42445.66, LD12

MOVR    -2741.25, AC1
*R      VD100, AC1
+R      AC1, LD12

MOVR    -194.433, AC1
*R      LD4, AC1
+R      AC1, LD12

MOVR    -23.49643, AC1
*R      LD0, AC1
+R      AC1, LD12

Network 13
// b = -35102.25650 + 610.9072320*l_dz -
    320.3346896*l_dy - 37251.63490*angles[0] +
    3950.337630*theta_2;
LD      SM0.0
MOVR    -35102.26, LD16

MOVR    610.9072, AC1
*R      LD4, AC1
+R      AC1, LD16

MOVR    -320.3347, AC1
*R      LD0, AC1
+R      AC1, LD16

MOVR    -37251.64, AC1
*R      VD100, AC1
+R      AC1, LD16

MOVR    3950.338, AC1
*R      LD8, AC1
+R      AC1, LD16

Network 14
// c = l_dz*l_dz + l_dy*l_dy - 6750.544760*
    theta_2 - 54978.00727 + 65801.36756*angles[0]
     + 232.600*l_dy*angles[0] - 24.675*l_dy*
    theta_2 - 464.4438236*l_dz + 62.9731176*l_dy;
LD      SM0.0
MOVR    LD4, LD20
*R      LD4, LD20

MOVR    LD0, AC1
*R      AC1, AC1
+R      AC1, LD20

MOVR    -6750.545, AC1
*R      LD8, AC1
+R      AC1, LD20

+R      -54978.01, LD20

MOVR    65801.37, AC1
*R      VD100, AC1
+R      AC1, LD20

MOVR    232.6, AC1
*R      LD0, AC1
*R      VD100, AC1
+R      AC1, LD20

MOVR    -24.675, AC1
*R      LD0, AC1
*R      LD8, AC1
+R      AC1, LD20

//- 464.4438236*l_dz + 62.9731176*l_dy;
MOVR    -464.4438, AC1
*R      LD4, AC1
+R      AC1, LD20

MOVR    62.97318, AC1
*R      LD0, AC1
+R      AC1, LD20

Network 15
// end Else
SCRE
Network 16
// phi = (-b - sqrt(b*b - 4*a*c)) / 2 / a;
LD      SM0.0
MOVR    LD16, AC1
AENO
*R      LD16, AC1
AENO
MOVR    LD12, AC2
AENO
*R      LD20, AC2
AENO
*R      -4.0, AC2
AENO
+R      AC2, AC1
AENO
SQRT    AC1, AC1
AENO
+R      LD16, AC1
AENO
/R      -2.0, AC1
AENO
MOVR    AC1, VD148
/R      LD12, VD148
Network 17
// phi_2 = phi * phi
LD      SM0.0
MOVR    VD148, VD152
*R      VD148, VD152
Network 18
// reserved
END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK cal_angle:SBR3
TITLE=void calculate_angle (int i);
VAR_INPUT
i:DINT;
END_VAR
VAR
l_dy:REAL;
l_dz:REAL;
aa:REAL;
bb:REAL;
cc:REAL;
END_VAR
BEGIN
Network 1
// l_dy = cosine_table[i]*n_dy + sine_table[i]*
    n_dz;
LD      SM0.0
SLD     LD0, 2
MOVD    VD172, AC1
+D      LD0, AC1
MOVR    VD140, LD4
*R      *AC1, LD4

MOVD    VD168, AC1
MOVR    VD144, AC2
+D      LD0, AC1
```

```
*R      *AC1, AC2
+R      AC2, LD4
Network 2
// l_dz = -sine_table[i]*n_dy + cosine_table[i]*
    n_dz;
LD      SM0.0
MOVD    VD172, AC1
+D      LD0, AC1
MOVR    VD144, LD8
*R      *AC1, LD8

MOVD    VD168, AC1
MOVR    VD140, AC2
+D      LD0, AC1
*R      *AC1, AC2
-R      AC2, LD8

Network 3
// if angles[0] < 4
LDR<    VD100, 4.0
=       S0.1
Network 4
// Then
LSCR    S0.1
Network 5
// a = 5046.043470 - 2398.594100*phi + 14.950*
    l_dy - 176.2232400*phi_2;
LD      SM0.0
MOVR    5046.043, LD12

MOVR    VD148, AC1
*R      -2398.594, AC1
+R      AC1, LD12

MOVR    LD4, AC1
*R      14.95, AC1
+R      AC1, LD12

MOVR    VD152, AC1
*R      -176.2232, AC1
+R      AC1, LD12

Network 6
// b = -34.750*l_dy + 5565.717330*phi -
    15246.68803 + 410.2085420*phi_2;
LD      SM0.0
MOVR    -34.75, LD16
*R      LD4, LD16

MOVR    VD148, AC1
*R      5565.717, AC1
+R      AC1, LD16

-R      15246.69, LD16

MOVR    VD152, AC1
*R      410.2086, AC1
+R      AC1, LD16

Network 7
// c = 610.9072320*l_dz*phi - 194.4329748*l_dz*
    phi_2 - 23.4964320*l_dy*phi_2 - 320.3346896*
    l_dy*phi + 87623.30094 + l_dz*l_dz + l_dy*
    l_dy + 36997.91703*phi_2 - 109037.6959*phi +
    524.6231176*l_dy - 464.4438236*l_dz;
LD      SM0.0
MOVR    610.9072, LD20
*R      LD8, LD20
*R      VD148, LD20

MOVR    -194.433, AC1
*R      LD8, AC1
*R      VD152, AC1
+R      AC1, LD20

MOVR    -23.49643, AC1
*R      LD4, AC1
*R      VD152, AC1
+R      AC1, LD20

MOVR    -320.3347, AC1
*R      LD4, AC1
*R      VD148, AC1
+R      AC1, LD20

+R      87623.3, LD20

MOVR    LD8, AC1
*R      AC1, AC1
+R      AC1, LD20

MOVR    LD4, AC1
*R      AC1, AC1
+R      AC1, LD20

//36997.91703*phi_2
MOVR    36997.92, AC1
*R      VD152, AC1
+R      AC1, LD20

//- 109037.6959*phi
MOVR    -109037.7, AC1
*R      VD148, AC1
+R      AC1, LD20

//524.6231176*l_dy
MOVR    524.6231, AC1
*R      LD4, AC1
+R      AC1, LD20

//- 464.4438236*l_dz
MOVR    -464.4438, AC1
*R      LD8, AC1
+R      AC1, LD20

Network 8
// end Then
SCRE
Network 9
// reverse if_angle
LDN     S0.1
=       S0.1
Network 10
// begin Else
LSCR    S0.1
Network 11
// a = -24.675*l_dy - 6750.544760 + 3950.337630*
    phi + 290.7683460*phi_2;
LD      SM0.0
MOVR    -24.675, LD12
*R      LD4, LD12

-R      6750.545, LD12

MOVR    3950.338, AC1
*R      VD148, AC1
+R      AC1, LD12

MOVR    290.7683, AC1
*R      VD152, AC1
+R      AC1, LD12

Network 12
// b = 232.600*l_dy + 65801.36756 - 37251.63490*
    phi - 2741.250400*phi_2;
LD      SM0.0
MOVR    232.6, LD16
*R      LD4, LD16

+R      65801.37, LD16
```

```
MOVR   -37251.64, AC1
*R     VD148, AC1
+R     AC1, LD16

MOVR   -2741.25, AC1
*R     VD152, AC1
+R     AC1, LD16

Network 13
// c = -194.4329748*l_dz*phi_2 - 23.4964320*l_dy*
    phi_2 - 320.3346896*l_dy*phi - 54978.00727 +
    l_dz*l_dz + l_dy*l_dy + 62.9731176*l_dy +
    42445.66269*phi_2 - 35102.25650*phi -
    464.4438236*l_dz + 610.9072320*l_dz*phi;
LD     SM0.0
//-194.4329748*l_dz*phi_2
MOVR   -194.433, LD20
*R     LD8, LD20
*R     VD152, LD20

// - 23.4964320*l_dy*phi_2
MOVR   -23.49643, AC1
*R     LD4, AC1
*R     VD152, AC1
+R     AC1, LD20

//- 320.3346896*l_dy*phi
MOVR   -320.3347, AC1
*R     LD4, AC1
*R     VD148, AC1
+R     AC1, LD20

//- 54978.00727
-R     54978.01, LD20

// l_dy^2 + l_dz^2
MOVR   LD8, AC1
*R     AC1, AC1
+R     AC1, LD20
MOVR   LD4, AC1
*R     AC1, AC1
+R     AC1, LD20

//62.9731176*l_dy
MOVR   62.97312, AC1
*R     LD4, AC1
+R     AC1, LD20

//42445.66269*phi_2
MOVR   42445.66, AC1
*R     VD152, AC1
+R     AC1, LD20

//- 35102.25650*phi
MOVR   -35102.26, AC1
*R     VD148, AC1
+R     AC1, LD20

//- 464.4438236*l_dz
MOVR   464.4438, AC1
*R     LD8, AC1
+R     AC1, LD20


//610.9072320*l_dz*phi
MOVR   610.9072, AC1
*R     LD8, AC1
*R     VD148, AC1
+R     AC1, LD20

Network 14
// end Else
SCRE
Network 15
// angles[i] = (-b + sqrt(b*b - 4*a*c)) / 2 / a;
    average_angle = average_angle * 0.9 + angles[
    i] * 0.1
LD     SM0.0
// calculate vane angle
// calculate b^2
MOVR   LD16, AC1
*R     AC1, AC1

// calculate -4ac
MOVR   -4.0, AC2
*R     LD12, AC2
*R     LD20, AC2

// calculate square root
+R     AC1, AC2
SQRT   AC2, AC1

// minus b
-R     LD16, AC1

// finalize result in AC1
/R     2.0, AC1
/R     LD12, AC1

// store result into angles[i]
MOVD   VD160, AC2
+D     LD0, AC2
MOVR   AC1, *AC2

// update average angle
MOVR   VD156, AC2
*R     0.9, AC2
*R     0.1, AC1
+R     AC1, AC2
MOVR   AC2, VD156

Network 16

END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK PID0_INIT:SBR4
TITLE=This POU was created by the PID formula of
    the S7-200 Instruction Wizard.
// To enable this configuration within the
    program, use SM0.0 to call this Subroutine
    from the MAIN program block every scan cycle.
      This code configures PID 0.  See DB1 for
    the PID loop variable table starting at VB620
    .  This subroutine initializes the variables
    used by the PID control logic and starts the
    PID Interrupt "PID_EXE" routine. The PID
    interrupt routine is called cyclically based
    on the PID sample time. For a complete
    description of the PID instruction see the S7
    -200 System Manual. Note:When the PID is in
    manual mode the output should be controlled
    by writing a normalized value(0.00 to 1.00)
    to the Manual Output parameter instead of
    changing the output directly. This will
    automatically provide a bumpless transfer
    when the PID is returned to automatic mode.
VAR_INPUT
PV_I:INT;       // Process Variable Input: Range
    2185 to 8266
Setpoint_R:REAL;        // Setpoint Input: Range
    0.0 to 100.0
END_VAR
VAR_OUTPUT
Output:INT;     // PID Output: Range 14000 to
    18000
END_VAR
VAR
Tmp_DI:DWORD;
Tmp_R:REAL;
END_VAR
```

```
BEGIN
Network 1 // This POU is password-protected
    against editing and viewing.

END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK S_RTI:SBR5
TITLE=LIBARY: Scale V1.2 (bipolar scaling)
// ============================
//
// LIABILITY
// Siemens AG does not accept liability of any
    kind for damages arising from the use of this
     application, except where it is obliged to
    by law, in cases such as damage to items used
     for personal purposes, personal injury,
    willful damage or gross negligence.
//
// WARRANTY
// The program examples given are specific
    solutions to complex tasks which were worked
    on by Customer Support. We must also point
    out that it is not possible in the current
    state of the technology to exclude all errors
     in software programs under all conditions of
     use. The program examples were prepared
    according to the best of our knowledge.
    However, we cannot accept any liability
    beyond the standard guarantee for Class C
    software in accordance with our General Terms
     of Sale for Software Products for Automation
     and Drive Technology". The program examples
    can be purchased on the Internet as single
    licenses. They may not be transferred to a
    third party.
//
// PASSWORD
// The password of the library is "1234"
//
// S_RTI
// Scale Real to Integer
//
//  The formula is as follows:
//  Ov = [(OSH - OSL) * (Iv - ISL) / (ISH - ISL)]
     + OSL
//  with ISL <= Iv <= ISH
//  and OSL <= Ov <= OSH
//
//  Ov  =       output value (INT)
//  Iv  =       input value (REAL)
//  OSH =       high limit of the scale for the
    output value (INT)
//  OSL =       low limit of the scale for the
    output value (INT)
//  ISH =       high limit of the scale for the
    input value (REAL)
//  ISL =       low limit of the scale for the
    input value (REAL)
//
//
//
//
// BIBLIOTHEK: Scale V1.2 (bipolare Skalierung)
// ====================================
//
// HAFTUNGSAUSSCHLUSS
// Bei diesem Programmbaustein handelt es sich um
     FREEWARE.
// Jedem Benutzer steht es frei, dieses Programm
    UNENTGELTLICH zu nutzen, zu kopieren und
    weiterzugeben.
// Die Autoren und Rechtsinhaber dieses Programms
     schlieen jegliche Haftung f r die
    Funktionstchtigkeit oder Kompatibilitt
    dieser Software aus.
// Die Benutzung erfolgt auf eigene Gefahr.
// Da diese Software kostenlos ist, entfllt
    jegliche Gewhrleistung, Anspruch auf
    Fehlerkorrektur und Hotlinesupport.
//
// PASSWORT
// Das Passwort der Bibliothek ist "1234"
//
// S_RTI
// Realzahl in ganze Zahl skalieren
//
//  Die Formel lautet wie folgt:
//  Ov = [(OSH - OSL) * (Iv - ISL) / (ISH - ISL)]
     + OSL
//  mit ISL <= Iv <= ISH
//  und OSL <= Ov <= OSH
//
//  Ov  =       Ausgangswert (INT)
//  Iv  =       Eingangswert (REAL)
//  OSH =       oberer Grenzwert der Skala f r
    den Ausgangswert (INT)
//  OSL =       unterer Grenzwert der Skala f r
    den Ausgangswert (INT)
//  ISH =       oberer Grenzwert der Skala f r
    den Eingangswert (REAL)
//  ISL =       unterer Grenzwert der Skala f r
    den Eingangswert (REAL)
VAR_INPUT
Input:REAL;
ISH:REAL;
ISL:REAL;
OSH:INT;
OSL:INT;
END_VAR
VAR_OUTPUT
Output:INT;
END_VAR
VAR
OSH_DI:DINT;
OSH_R:REAL;
temp_I:INT;
OSL_DI:DINT;
END_VAR
BEGIN
Network 1 // This POU is password-protected
    against editing and viewing.

END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK set:SBR6
TITLE=set values of the motions.
BEGIN
Network 1 // step
// stop invalid values
LD      I1.0
LDW<    VW228, 0
OW>     VW228, 80
ALD
R       I1.0, 1
Network 2
// update parameters
LD      I1.0
DTCH    11
AENO
MOVW    VW228, VW218
AENO
R       I1.0, 1
R       M0.0, 1
Network 3
// carry out: step
LDN     M0.0
CALL    SBR1, VW218, 80, 0, 0.0, 100.0, VD204
Network 4
// bound check: begin sine output
LD      I1.1
LDW<    VW224, 0
OW>     VW224, 80
```

```
OW<     VW222, 0
OW>     VW222, 80
OW<     VW220, 0
OW>     VW220, 1000
ALD
R       I1.1, 1
Network 5
// execute sine output: implemented using timer
    interrupt 1
LD      I1.1
DTCH    11
AENO
MOVW    VW220, VW216
AENO
*I      +20, VW216
CALL    SBR1, VW224, 80, 0, 0.0, 100.0, VD310
CALL    SBR1, VW222, 80, 0, 0.0, 100.0, VD314
MOVB    1, SMB35
AENO
MOVW    0, VW212
AENO
MOVW    0, VW214
AENO

ATCH    INT0, 11
AENO
R       I1.1, 1
S       M0.0, 1
END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK init:SBR7
TITLE=Initialize pointers
BEGIN
Network 1
// This is necessary because "&" cannot be used
    in data block. Best if other solution is
    found.
LD      SM0.0
MOVD    &VD0, VD168
MOVD    &VD40, VD172
MOVD    &VD80, VD176
MOVD    &VD100, VD160
MOVD    &AIW0, VD184
Network 2 // Time setting
LD      V308.0
CALL    SBR8, &VB300
Network 3
LD      SM0.0
MOVR    50.0, VD204
END_SUBROUTINE_BLOCK
SUBROUTINE_BLOCK SET_RTC_I:SBR8
TITLE=NAME:       Clock_INT
// VERSION: 1.0
// DATE:          07/2004
//
//
    ##########################################################

//  LICENSE:
//  This program is distributed as freeware.
//
//  Siemens makes no warranty, expressed or
    implied, with regard to this software.
//  All implied warranties, including the
    warranties of merchantability and fitness for
     a
//  particular use, are hereby excluded.
//
//  Under no circumstances shall Siemens, or the
    authors of this product,
//  be liable for any incidental or consequential
     damages, nor for any damages.
//
    ##########################################################

//
```

```
// This program is protected against
    unintentional delete.
// If you wish to modify the routine
// PASSWORD = EDIT
//
// This routine simplify the usage of the
    MicroWin shipped "SET_RTC" function.
// So the user doesn't need to convert the data
    to BCD format before deliver to the MicroWin
    "SET_RTC" function.
//
// This routine will read the user values and
    convert them from integer to BCD format and
// set the clock with these values.
//
// Requirements:
// - CPU with clock (CPU221,CPU222 have not a
    internal clock, external clock module is
    necessary)
// - MicroWin V3.2 SP4 or higher
// - Program memory space:
// - SET_RTC_I  126 Byte
// - Full Library     236 Byte
//
// The program has 1 input and 0 output.
//
// NOTICE:
// The program dosent check the validity of the
    values.
//
//  Parameter
//  Address      This must be an address of a V-
    memory location. This address is used as the
    first byte of a table which
//                    has a length of 8 Bytes.
    Please ensure that the 8 byte behind are not
    used with other program parts.
//
//  Example:
//                                  _____________
//                  I   SET_RTC_I   I
//                                  I
                                    I
//  I------I SM0.0 I-----I EN                    I
//                                  I
                                    I
//                                  I
                                    I
//                &VB0 --I Address             I
//                                  I_____________I
//
VAR_INPUT
Address:DWORD;  // Table address where the set
    value are deposited
END_VAR
VAR
INDX:WORD;      // loop counter
Adr_Mem_1:DWORD;        // temp memory  address
Adr_Mem_2:DWORD;        // temp memory  address
tmp_W:WORD;     // temp memory
END_VAR
BEGIN
Network 1 // This POU is password-protected
    against editing and viewing.

END_SUBROUTINE_BLOCK
INTERRUPT_BLOCK INT_0:INT0
TITLE=Timer interrupt 1, used to implement sine
    output
BEGIN
Network 1
// Increment clock tick (1ms)
LD      SM0.0
INCW    VW212
Network 2
```

```
// Check parameters and update command value
LDW>=  VW212, VW216
LPS
MOVW   0, VW212
AENO
ITD    VW214, AC1
AENO
SLD    AC1, 2
AENO
MOVD   &VD1000, AC2
AENO
+D     AC2, AC1
AENO
CALL   SBR1, *AC1, 1, -1, VD314, VD310, VD204
LPP
INCW   VW214
AENO
AW>=   VW214, 50
MOVW   0, VW214
END_INTERRUPT_BLOCK
INTERRUPT_BLOCK PID_EXE:INT1
TITLE=This POU was created by the PID formula of
    the S7-200 Instruction Wizard.
// This interrupt routine implements Timed
    Interrupt for PID execution. This interrupt
    routine was attached in subroutine "PID0_INIT
    ".
BEGIN
Network 1 // This POU is password-protected
    against editing and viewing.

END_INTERRUPT_BLOCK
```

```
SUBROUTINE_BLOCK init:SBR7
TITLE=Initialize pointers
BEGIN
Network 1
// This is necessary because "&" cannot be used
    in data block. Best if other solution is
    found.
LD     SM0.0
MOVD   &VD0, VD168
MOVD   &VD40, VD172
MOVD   &VD80, VD176
MOVD   &VD100, VD160
MOVD   &AIW0, VD184
Network 2 // Time setting
LD     V308.0
CALL   SBR8, &VB300
Network 3
LD     SM0.0
MOVR   50.0, VD204
END_SUBROUTINE_BLOCK
```

```
SUBROUTINE_BLOCK read_inputs:SBR0
TITLE=void read_inputs ();
VAR
temp:REAL;
END_VAR
BEGIN
Network 1
// n_dy = (float)d_sensors[0]/full_scale_f -
    disp_offsets[0];
LD     SM0.0
MOVR   VD164, AC1
AENO
+R     *VD176, AC1
AENO
CALL   SBR1, *VD184, VW180, 0, AC1, *VD176, VD140
Network 2
// n_dz = (float)d_sensors[2]/full_scale_f -
    disp_offsets[2];
LD     SM0.0
MOVD   VD184, AC1
AENO
+D     +4, AC1
AENO
MOVD   VD176, AC2
AENO
+D     +8, AC2
AENO
MOVR   VD164, AC3
AENO
+R     *AC2, AC3
AENO
CALL   SBR1, *AC1, VW180, 0, AC3, *AC2, VD144
Network 3
// n_dy = (n_dz - n_dy)/2;
LD     SM0.0
MOVR   VD144, AC1
AENO
-R     VD140, AC1
AENO
MOVR   AC1, VD140
/R     2.0, VD140
Network 4
// temp = (float)d_sensors[1]/full_scale_f -
    disp_offsets[1];
LD     SM0.0
MOVD   VD184, AC1
AENO
+D     +2, AC1
AENO
MOVD   VD176, AC2
AENO
+D     +4, AC2
AENO
MOVR   VD164, AC3
AENO
+R     *AC2, AC3
AENO
CALL   SBR1, *AC1, VW180, 0, AC3, *AC2, LD0

Network 5
// n_dz = (float)d_sensors[3]/full_scale_f -
    disp_offsets[3];
LD     SM0.0
MOVD   VD184, AC1
AENO
+D     +6, AC1
AENO
MOVD   VD176, AC2
AENO
+D     +12, AC2
AENO
MOVR   VD164, AC3
AENO
+R     *AC2, AC3
AENO
CALL   SBR1, *AC1, VW180, 0, AC3, *AC2, VD144
Network 6
// n_dz = (n_dz - temp)/1.932 + n_dy/3.732;
LD     SM0.0
MOVR   VD144, AC1
AENO
-R     LD0, AC1
AENO
/R     1.932, AC1
AENO
MOVR   VD140, AC2
AENO
/R     3.732, AC2
AENO
MOVR   AC1, VD144
+R     AC2, VD144
Network 7
// angles[0] = (float)a_sensor/full_scale_f -
    angle_offset;
LD     SM0.0
MOVR   VD96, AC1
```

```
AENO
+R      VD188, AC1
AENO
CALL    SBR1, AIW8, VW180, 0, AC1, VD96, *VD160
Network 8
// average_angle = average_angle * 0.9 + angles
    [0] * 0.1;
LD      SM0.0
MOVR    VD156, AC1
AENO
*R      0.9, AC1
AENO
MOVR    *VD160, AC2
AENO
*R      0.1, AC2
AENO
MOVR    AC1, VD156
+R      AC2, VD156
END_SUBROUTINE_BLOCK
```

```
SUBROUTINE_BLOCK set:SBR6
TITLE=set values of the motions.
BEGIN
Network 1 // step
// stop invalid values
LD      I1.0
LDW<    VW228, 0
OW>     VW228, 80
ALD
R       I1.0, 1
Network 2
// update parameters
LD      I1.0
DTCH    11
AENO
MOVW    VW228, VW218
AENO
R       I1.0, 1
R       M0.0, 1
Network 3
// carry out: step
LDN     M0.0
CALL    SBR1, VW218, 80, 0, 0.0, 100.0, VD204
Network 4
// bound check: begin sine output
LD      I1.1
LDW<    VW224, 0
OW>     VW224, 80
OW<     VW222, 0
OW>     VW222, 80
OW<     VW220, 0
OW>     VW220, 1000
ALD
R       I1.1, 1
Network 5
// execute sine output: implemented using timer
    interrupt 1
LD      I1.1
DTCH    11
AENO
MOVW    VW220, VW216
AENO
*I      +20, VW216
CALL    SBR1, VW224, 80, 0, 0.0, 100.0, VD310
CALL    SBR1, VW222, 80, 0, 0.0, 100.0, VD314
MOVB    1, SMB35
AENO
MOVW    0, VW212
AENO
MOVW    0, VW214
AENO

ATCH    INT0, 11
AENO
R       I1.1, 1
S       M0.0, 1
END_SUBROUTINE_BLOCK
```

```
SUBROUTINE_BLOCK cal_phi:SBR2
TITLE=void calculate_phi ();
VAR
l_dy:REAL;
l_dz:REAL;
theta_2:REAL;
aa:REAL;
bb:REAL;
cc:REAL;
END_VAR
BEGIN
Network 1 // Network Title
// l_dy = 0.9962*n_dy + 0.08716*n_dz;
LD      SM0.0
MOVR    0.9962, LD0
*R      VD140, LD0

MOVR    0.08716, AC1
*R      VD144, AC1
+R      AC1, LD0

Network 2
// l_dz = -0.08716*n_dy + 0.9962*n_dz;
LD      SM0.0
MOVR    -0.08716, LD4
*R      VD140, LD4

MOVR    0.9962, AC1
*R      VD144, AC1
+R      AC1, LD4

Network 3
// theta_2 = angles[0] * angles[0];
LD      SM0.0
MOVR    VD100, LD8
*R      VD100, LD8

Network 4
// if angles[0] < 4
LDR<    VD100, 4.0
=       S0.0
Network 5
// then
LSCR    S0.0
Network 6
// a = -176.223*theta_2 - 23.496*l_dy + 36997.917
     + 410.20*angles[0] - 194.433*l_dz;
LD      SM0.0
MOVR    -176.233, LD12
*R      LD8, LD12

MOVR    -23.496, AC1
*R      LD0, AC1
+R      AC1, LD12

+R      36997.92, LD12

MOVR    410.2, AC1
*R      VD100, AC1
+R      AC1, LD12

MOVR    -194.433, AC1
*R      LD4, AC1
+R      AC1, LD12

Network 7
// b = -109037.6959 + 610.9072320*l_dz -
    320.3346896*l_dy + 5565.717330*angles[0] -
    2398.594100*theta_2;
LD      SM0.0
MOVR    610.9072, LD16
*R      LD4, LD16
```

```
+R     -109037.7, LD16

MOVR   LD0, AC1
*R     -320.3347, AC1
+R     AC1, LD16

MOVR   5565.717, AC1
*R     VD100, AC1
+R     AC1, LD16

MOVR   -2398.594, AC1
*R     LD8, AC1
+R     AC1, LD16

Network 8
// c = 87623.3 + l_dz*l_dz + l_dy*l_dy +
     5046.043470*theta_2 + 14.950*l_dy*theta_2 -
     464.4438236*l_dz
// - 15246.68803*angles[0] - 34.750*l_dy*angles
     [0] + 524.6231176*l_dy;
LD     SM0.0
MOVR   LD4, LD20
*R     LD4, LD20

MOVR   LD0, AC1
*R     AC1, AC1
+R     AC1, LD20

+R     87623.3, LD20

MOVR   5046.043, AC1
*R     LD8, AC1
+R     AC1, LD20

MOVR   14.95, AC1
*R     LD0, AC1
*R     LD8, AC1
+R     AC1, LD20

//- 464.4438236*l_dz
MOVR   -464.4438, AC1
*R     LD4, AC1
+R     AC1, LD20

//- 15246.68803*angles[0]
MOVR   -15246.69, AC1
*R     VD100, AC1
+R     AC1, LD20

//- 34.750*l_dy*angles[0]
MOVR   -34.75, AC1
*R     LD0, AC1
*R     VD100, AC1
+R     AC1, LD20

//+ 524.6231176*l_dy
MOVR   524.6231, AC1
*R     LD0, AC1
+R     AC1, LD20

Network 9
// end Then;
SCRE
Network 10
// reverse if_phi
LDN    S0.0
=      S0.0
Network 11
// begin Else
LSCR   S0.0
Network 12
// a = 290.7683460*theta_2 + 42445.66269 -
     2741.250400*angles[0] - 194.4329748*l_dz -
     23.4964320*l_dy;
LD     SM0.0
MOVR   290.7683, LD12
*R     LD8, LD12

+R     42445.66, LD12

MOVR   -2741.25, AC1
*R     VD100, AC1
+R     AC1, LD12

MOVR   -194.433, AC1
*R     LD4, AC1
+R     AC1, LD12

MOVR   -23.49643, AC1
*R     LD0, AC1
+R     AC1, LD12

Network 13
// b = -35102.25650 + 610.9072320*l_dz -
     320.3346896*l_dy - 37251.63490*angles[0] +
     3950.337630*theta_2;
LD     SM0.0
MOVR   -35102.26, LD16

MOVR   610.9072, AC1
*R     LD4, AC1
+R     AC1, LD16

MOVR   -320.3347, AC1
*R     LD0, AC1
+R     AC1, LD16

MOVR   -37251.64, AC1
*R     VD100, AC1
+R     AC1, LD16

MOVR   3950.338, AC1
*R     LD8, AC1
+R     AC1, LD16

Network 14
// c = l_dz*l_dz + l_dy*l_dy - 6750.544760*
     theta_2 - 54978.00727 + 65801.36756*angles[0]
      + 232.600*l_dy*angles[0] - 24.675*l_dy*
     theta_2 - 464.4438236*l_dz + 62.9731176*l_dy;
LD     SM0.0
MOVR   LD4, LD20
*R     LD4, LD20

MOVR   LD0, AC1
*R     AC1, AC1
+R     AC1, LD20

MOVR   -6750.545, AC1
*R     LD8, AC1
+R     AC1, LD20

+R     -54978.01, LD20

MOVR   65801.37, AC1
*R     VD100, AC1
+R     AC1, LD20

MOVR   232.6, AC1
*R     LD0, AC1
*R     VD100, AC1
+R     AC1, LD20

MOVR   -24.675, AC1
*R     LD0, AC1
*R     LD8, AC1
+R     AC1, LD20

//- 464.4438236*l_dz + 62.9731176*l_dy;
```

```
MOVR   -464.4438, AC1
*R     LD4, AC1
+R     AC1, LD20

MOVR   62.97318, AC1
*R     LD0, AC1
+R     AC1, LD20

Network 15
// end Else
SCRE
Network 16
// phi = (-b - sqrt(b*b - 4*a*c)) / 2 / a;
LD     SM0.0
MOVR   LD16, AC1
AENO
*R     LD16, AC1
AENO
MOVR   LD12, AC2
AENO
*R     LD20, AC2
AENO
*R     -4.0, AC2
AENO
+R     AC2, AC1
AENO
SQRT   AC1, AC1
AENO
+R     LD16, AC1
AENO
/R     -2.0, AC1
AENO
MOVR   AC1, VD148
/R     LD12, VD148
Network 17
// phi_2 = phi * phi
LD     SM0.0
MOVR   VD148, VD152
*R     VD148, VD152
Network 18
// reserved
END_SUBROUTINE_BLOCK
```

```
SUBROUTINE_BLOCK cal_angle:SBR3
TITLE=void calculate_angle (int i);
VAR_INPUT
i:DINT;
END_VAR
VAR
l_dy:REAL;
l_dz:REAL;
aa:REAL;
bb:REAL;
cc:REAL;
END_VAR
BEGIN
Network 1
// l_dy = cosine_table[i]*n_dy + sine_table[i]*
    n_dz;
LD     SM0.0
SLD    LD0, 2
MOVD   VD172, AC1
+D     LD0, AC1
MOVR   VD140, LD4
*R     *AC1, LD4

MOVD   VD168, AC1
MOVR   VD144, AC2
+D     LD0, AC1
*R     *AC1, AC2
+R     AC2, LD4
Network 2
// l_dz = -sine_table[i]*n_dy + cosine_table[i]*
    n_dz;
LD     SM0.0
MOVD   VD172, AC1
+D     LD0, AC1
MOVR   VD144, LD8
*R     *AC1, LD8

MOVD   VD168, AC1
MOVR   VD140, AC2
+D     LD0, AC1
*R     *AC1, AC2
-R     AC2, LD8

Network 3
// if angles[0] < 4
LDR<   VD100, 4.0
=      S0.1
Network 4
// Then
LSCR   S0.1
Network 5
// a = 5046.043470 - 2398.594100*phi + 14.950*
    l_dy - 176.2232400*phi_2;
LD     SM0.0
MOVR   5046.043, LD12

MOVR   VD148, AC1
*R     -2398.594, AC1
+R     AC1, LD12

MOVR   LD4, AC1
*R     14.95, AC1
+R     AC1, LD12

MOVR   VD152, AC1
*R     -176.2232, AC1
+R     AC1, LD12

Network 6
// b = -34.750*l_dy + 5565.717330*phi -
    15246.68803 + 410.2085420*phi_2;
LD     SM0.0
MOVR   -34.75, LD16
*R     LD4, LD16

MOVR   VD148, AC1
*R     5565.717, AC1
+R     AC1, LD16

-R     15246.69, LD16

MOVR   VD152, AC1
*R     410.2086, AC1
+R     AC1, LD16

Network 7
// c = 610.9072320*l_dz*phi - 194.4329748*l_dz*
    phi_2 - 23.4964320*l_dy*phi_2 - 320.3346896*
    l_dy*phi + 87623.30094 + l_dz*l_dz + l_dy*
    l_dy + 36997.91703*phi_2 - 109037.6959*phi +
    524.6231176*l_dy - 464.4438236*l_dz;
LD     SM0.0
MOVR   610.9072, LD20
*R     LD8, LD20
*R     VD148, LD20

MOVR   -194.433, AC1
*R     LD8, AC1
*R     VD152, AC1
+R     AC1, LD20

MOVR   -23.49643, AC1
*R     LD4, AC1
*R     VD152, AC1
+R     AC1, LD20

MOVR   -320.3347, AC1
```

```
*R      LD4, AC1
*R      VD148, AC1
+R      AC1, LD20

+R      87623.3, LD20

MOVR    LD8, AC1
*R      AC1, AC1
+R      AC1, LD20

MOVR    LD4, AC1
*R      AC1, AC1
+R      AC1, LD20

//36997.91703*phi_2
MOVR    36997.92, AC1
*R      VD152, AC1
+R      AC1, LD20

//- 109037.6959*phi
MOVR    -109037.7, AC1
*R      VD148, AC1
+R      AC1, LD20

//524.6231176*l_dy
MOVR    524.6231, AC1
*R      LD4, AC1
+R      AC1, LD20

//- 464.4438236*l_dz
MOVR    -464.4438, AC1
*R      LD8, AC1
+R      AC1, LD20

Network 8
// end Then
SCRE
Network 9
// reverse if_angle
LDN     S0.1
=       S0.1
Network 10
// begin Else
LSCR    S0.1
Network 11
// a = -24.675*l_dy - 6750.544760 + 3950.337630*
    phi + 290.7683460*phi_2;
LD      SM0.0
MOVR    -24.675, LD12
*R      LD4, LD12

-R      6750.545, LD12

MOVR    3950.338, AC1
*R      VD148, AC1
+R      AC1, LD12

MOVR    290.7683, AC1
*R      VD152, AC1
+R      AC1, LD12

Network 12
// b = 232.600*l_dy + 65801.36756 - 37251.63490*
    phi - 2741.250400*phi_2;
LD      SM0.0
MOVR    232.6, LD16
*R      LD4, LD16

+R      65801.37, LD16

MOVR    -37251.64, AC1
*R      VD148, AC1
+R      AC1, LD16

MOVR    -2741.25, AC1
*R      VD152, AC1
+R      AC1, LD16

Network 13
// c = -194.4329748*l_dz*phi_2 - 23.4964320*l_dy*
    phi_2 - 320.3346896*l_dy*phi - 54978.00727 +
    l_dz*l_dz + l_dy*l_dy + 62.9731176*l_dy +
    42445.66269*phi_2 - 35102.25650*phi -
    464.4438236*l_dz + 610.9072320*l_dz*phi;
LD      SM0.0
//-194.4329748*l_dz*phi_2
MOVR    -194.433, LD20
*R      LD8, LD20
*R      VD152, LD20

// - 23.4964320*l_dy*phi_2
MOVR    -23.49643, AC1
*R      LD4, AC1
*R      VD152, AC1
+R      AC1, LD20

//- 320.3346896*l_dy*phi
MOVR    -320.3347, AC1
*R      LD4, AC1
*R      VD148, AC1
+R      AC1, LD20

//- 54978.00727
-R      54978.01, LD20

// l_dy^2 + l_dz^2
MOVR    LD8, AC1
*R      AC1, AC1
+R      AC1, LD20
MOVR    LD4, AC1
*R      AC1, AC1
+R      AC1, LD20

//62.9731176*l_dy
MOVR    62.97312, AC1
*R      LD4, AC1
+R      AC1, LD20

//42445.66269*phi_2
MOVR    42445.66, AC1
*R      VD152, AC1
+R      AC1, LD20

//- 35102.25650*phi
MOVR    -35102.26, AC1
*R      VD148, AC1
+R      AC1, LD20

//- 464.4438236*l_dz
MOVR    464.4438, AC1
*R      LD8, AC1
+R      AC1, LD20


//610.9072320*l_dz*phi
MOVR    610.9072, AC1
*R      LD8, AC1
*R      VD148, AC1
+R      AC1, LD20

Network 14
// end Else
SCRE
Network 15
// angles[i] = (-b + sqrt(b*b - 4*a*c)) / 2 / a;
    average_angle = average_angle * 0.9 + angles[
    i] * 0.1
LD      SM0.0
// calculate vane angle
// calculate b^2
```

```
MOVR   LD16, AC1
*R     AC1, AC1

// calculate -4ac
MOVR   -4.0, AC2
*R     LD12, AC2
*R     LD20, AC2

// calculate square root
+R     AC1, AC2
SQRT   AC2, AC1

// minus b
-R     LD16, AC1

// finalize result in AC1
/R     2.0, AC1
/R     LD12, AC1

// store result into angles[i]
MOVD   VD160, AC2
+D     LD0, AC2
MOVR   AC1, *AC2

// update average angle
MOVR   VD156, AC2
*R     0.9, AC2
*R     0.1, AC1
+R     AC1, AC2
MOVR   AC2, VD156

Network 16

END_SUBROUTINE_BLOCK
```

```
INTERRUPT_BLOCK INT_0:INT0
TITLE=Timer interrupt 1, used to implement sine
    output
BEGIN
Network 1
// Increment clock tick (1ms)
LD     SM0.0
INCW   VW212
Network 2
// Check parameters and update command value
LDW>=  VW212, VW216
LPS
MOVW   0, VW212
AENO
ITD    VW214, AC1
AENO
SLD    AC1, 2
AENO
MOVD   &VD1000, AC2
AENO
+D     AC2, AC1
AENO
CALL   SBR1, *AC1, 1, -1, VD314, VD310, VD204
LPP
INCW   VW214
AENO
AW>=   VW214, 50
MOVW   0, VW214
END_INTERRUPT_BLOCK
```

Look-up tables and initialized program parameters are shown below.

```
DATA_BLOCK_TAB USER1
//
BEGIN
//
//DATA PAGE COMMENTS
//
//Press F1 for help and example data page

// const
VD0    0.08716, 0.656, 0.9744, 0.9205, 0.515,
    -0.08716, -0.656, -0.9744, -0.9205, -0.515 //
     [10] Sine table
VD40   0.9962, 0.7547, 0.225, -0.3907, -0.8572,
    -0.9962, -0.7547, -0.225, 0.3907, 0.8572 //
    [10] Cosine table
VD80   0.0, 0.0, 0.0, 0.0          // [4]
    displacement offset
VD96   0.0                         // angle
    offset
VW180  32000                       // full scale
    for sensor reading
VD164  25.0                        // actual
    scale for displacement sensor, unit: mm
VD188  6.195919
VB300  14, 7, 31, 13, 0, 0, 0, 5   // year, month
    , day, hour, minute, second, reserved, week
    day
VB308  2#1                         // update

// global
VD156  0                           // control
    feedback
VD1000 0.12533, 0.24869, 0.36812, 0.48176,
    0.58779, 0.68455, 0.77052, 0.84432, 0.90484,
    0.95106, 0.98229, 0.99803, 0.99803, 0.98229,
    0.95106, 0.90484, 0.84432, 0.77052, 0.68455,
    0.58779, 0.48176, 0.36812, 0.24869, 0.12533,
    0, -0.12533, -0.24869, -0.36812, -0.48176,
    -0.58779, -0.68455, -0.77052, -0.84432,
    -0.90484, -0.95106, -0.98229, -0.99803,
    -0.99803, -0.98229, -0.95106, -0.90484,
    -0.84432, -0.77052, -0.68455, -0.58779,
    -0.48176, -0.36812, -0.24869, -0.12533

END_DATA_BLOCK_TAB
DATA_BLOCK_TAB PID0_DATA
//
BEGIN
//
    ------------------------------------------------------

//The following were generated by the S7-200
    Instruction Wizard, PID Formula.
//Parameter Table for PID 0.
//
    ------------------------------------------------------

VD620  0.0                         //Process
    Variable
VD624  0.0                         //Loop
    Setpoint
VD628  0.0                         //Calculated
    Loop Output
VD632  2.0                         //Loop Gain
VD636  0.1                         //Sample Time
VD640  1.0                         //Integral
    Time
VD644  0.0                         //Derivative
    Time
VD648  0.0                         //Integral Sum
     or Bias
VD652  0.0                         //Value of
    Process Variable stored from last execution.
VB656  'PIDA'                      //Extended
    Loop Table Marker
VB660  16#00                       //Algorithm
    control byte
VB661  16#00                       //Algorithm
    status byte
VB662  16#00                       //Algorithm
    result byte
```

```
VB663  16#03                        //Algorithm
    configuration byte
VD664  0.08                         //Deviation
    value set from Advanced button or from
    default value
VD668  0.02                         //Hysteresis
    value set from Advanced button or from
    default value
VD672  0.1                          //Initial
    Output step value set from Advanced button or
     from default value
VD676  7200.0                       //Watchdog
    timeout value set from Advanced button or
    from default value
VD680  0.0                          //Gain value
    determined by Auto Tune algorithm
VD684  0.0                          //Integral
    time value determined by Auto Tune algorithm
VD688  0.0                          //Derivative
    time value determined by Auto Tune algorithm
VD692  0.0                          //Deviation
    value calculated by algorithm if automatic
    calculation option set
VD696  0.0                          //Hysteresis
    value calculated by algorithm if automatic
    calculation option set
END_DATA_BLOCK_TAB
```