POST-INSTALLABLE AUTOMATIC ADJUSTMENT DEVICE FOR AIR CONDITIONER

Bachelor's Thesis

Sakera Joarder

Degree Programme in Information Technology

Accepted ___.___._____ _____

SAVONIA UNIVERSITY OF APPLIED SCIENCES

Degree Programme

Information Technology

Author

Sakera Joarder

Title of Project

Post-Installable Automatic Adjustment Device for Air Conditioner

Type of Project	Date	Pages
Thesis	09 May 2011	55 + 11

Academic Supervisor	Company Supervisor
Mr Asmo Jakorinne	Mr Mikko Laasanen

Company

SAVONIA UNIVERSITY OF APPLIED SCIENCES

Abstract

The goal of the thesis was to develop a prototype of a post-installable device to adjust the air conditioner automatically according to the humidity and temperature readings. The idea is that the user can define the humidity and temperature limits easily based on the need and according to the defined limits the air conditioner is automatically adjusted. Some air conditioners nowadays have an automatic microcontroller or a computer based adjustment system but not all. The device is intended to be used in the garage especially during the winter. The same device can also be used e. g. to adjust the residential air conditioning system when using the washing room or sauna.

The project was carried out by using an RH&T sensor, a PC, LabVIEW software, a step motor and a step motor controller. The humidity and temperature values were read by a reading device created using an RH&T (Relative Humidity and Temperature) sensor. The programming part of the prototype was done via LabVIEW. The project was a success and accomplished without significant problems. The prototype worked in the required way and was tested in a real environment.

Keywords

Air Condtioner, Control System, LabVIEW, Step Motor, Step Motor Controller, RH&T sensor

Confidentiality

public

SAVONIA-AMMATTIKORKEAKOULU TEKNIIKKA KUOPIO

Koulutusohjelma

Information Technology

Tekijä

Sakera Joarder

Työn nimi

Post-Installable Automatic Adjustment Device for Air Conditioner

Työn laji	Päiväys	Sivumäärä
Opinnäytetyö	9.5.2011	55 + 11

Työn valvoja	Yrityksen yhdyshenkilö
Asmo Jakorinne	Mikko Laasanen

Yritys

Savonia-AMK

Tiivistelmä

Tämän opinnäytetyön aiheena oli kehittää prototyyppi automaattisäätölaitteesta ilmastointilaitteelle käyttäen askelmoottoria, joka säätää ilmastointilaitetta anturilta saatujen kosteus- ja lämpötila-arvojen perusteella. Tarkoitus oli, että käyttäjä pystyy tarpeen mukaan määrittelemään kosteuden ja lämpötilan raja-arvon, jonka perusteella ilmastointilaitetta säädetään automaattisesti. Monessa ilmastointilaitteessa nykyään on microcontrolleri- tai tietokoneohjauksella oleva automaattinen säätö. Laitetta aiotaan käyttää erityisesti autotallissa varsinkin talvella. Samaa laitetta voidaan myös käyttää esimerkiksi kodinhoitohuoneessa tai saunassa ilmastoinnin säätämiseen.

Opinnäytetyö toteutettiin käyttäen RH&T- kosteus- ja lämpötila-anturia, tietokonetta, LabVIEW- ohjelmointikieltä, askelmoottoria ja askelmoottorin ohjainta. Kosteus- ja lämpötila-arvot on luettu laitteella, joka rakennettiin käyttäen RH&T-anturia. Prototyypin ohjelmointiosuus toteutettiin kokonaan LabVIEW-ohjelmointikielellä. Opinnäytetyö onnistui ilman huomattavia ongelmia ja prototyyppiä testattiin oikeassa ympäristössä.

Avainsanat

Air Condtioner, Control System, LabVIEW, Step Motor, Step Motor Controller, RH&T sensor

Luottamuksellisuus

julkinen

Abbreviations

LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
RH&T	Relative Humidity and Temperature
NI	National Instruments
LV	LabVIEW
VI	Virtual Instrument
PIC	Peripheral Interface Controller
VISA	Visual Instrument Software Architecture
CAV	Constant Air Volume
VAV	Variable Air Volume
AUH	Air-Handling Unit

Boolean	Boolean is a logical data type returning a true or false value.
Enumerated	Enumerated is data type consisting of named values.

Contents

1 INTRODUCTION

The goal of the thesis is to develop a prototype of a post-installable device to adjust the air conditioner automatically according to the humidity and the temperature readings. Some air conditioners nowadays have an automatic microcontroller or computer based adjustment system but not all. The device is intended to be used in the garage especially during the winter. The same device can also be used e. g. to adjust the residential air conditioning system when using the washing room or sauna. This project is a single project and is ordered by Savonia University of Applied Sciences, Information Technology R&D Unit. The project is interesting because it is a nice entirety and intended to be accomplished by LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) graphical programming language. LabVIEW is a top class tool for data acquisition, instrument control, measurement and control, designing and prototyping of embedded systems in the industrial world and education. Many companies e.g. Honeywell, Medikro nowadays use LabVIEW in research & development and even as a user interface in commercial products.

This project requires good skills of the LabVIEW programming language and familiarization with air conditioning systems and mechanical device planning in order to make and install the device. The project will be carried out in the Savonia University of Applied Sciences, Information Technology R&D Unit and the device will be tested in a real environment.

2 PRINCIPLES OF AIR CONDITIONING SYSTEMS

Air conditioning systems are one type of mechanical ventilation systems. Air conditioning is the air circulation process to adjust heat, humidity and clean the indoor air in order to get thermal comfort and fresh air. Air conditioning systems can also be referred to any form of cooling, heating, ventilation, or disinfection that modifies the condition of air. Air conditioners (often referred to as AC or air con.) are an appliance, a system, or a machine designed to stabilise the air temperature and humidity within an area (used for cooling as well as heating depending on the air properties at a given time), typically using a refrigeration cycle but sometimes using evaporation, commonly for comfort cooling in buildings and motor vehicles. Air conditioning, including filtration, humidification, cooling, disinfection, etc., provides a clean, safe, hypoallergenic indoor atmosphere. Air conditioning can have a positive effect on sufferers of allergies and asthma. Conversely, if air conditioners are not kept clean it can increase the growth and spread of microorganisms. So, it is very important to keep air conditioners clean in order to avoid possible side effects and get the best advantage of them. [1]

2.1 Types of Air Conditioning Systems

Although air is not an effective heat transportation medium, but still is used to heat or cool and to control the humidity of indoor spaces. There are various techniques and systems that are used to accomplish these purposes. In this chapter different types of air conditioning systems are explained. [1]

2.1.1 Fan Coil and Induction Systems

In fan coil and induction air conditioning systems energy is transported as hot or chilled water in hydronic network to each space that are containing the hydronic network. Fresh air is supplied to the spaces and the air is extracted if appropriate. Each space containing the hydronic network has heat exhausters to cool and dry or heat the air. The air is

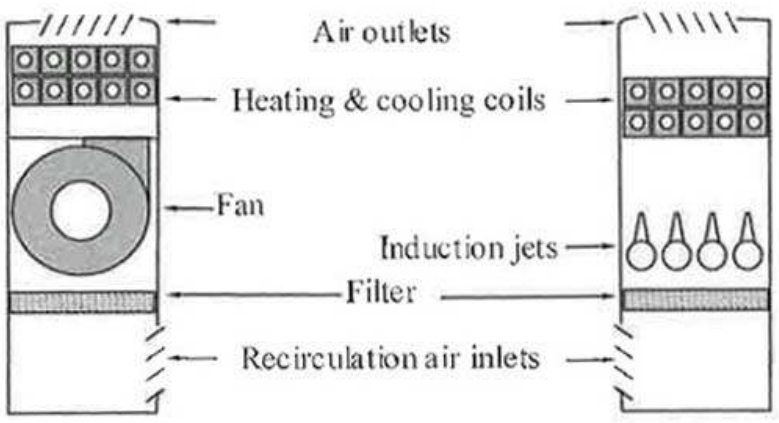


Figure 1. Schematic cut through fan coil and induction units. [1]

forced through these heat exhausters either by a fan coil units or by induction units. The schematic cut though a fan coil and induction units are shown in Figure 1. [1]

2.1.2 Constant and Variable Air Volume Systems

If heat is transported with air, the indoor temperature can either be controlled by varying the supplied air temperature (CAV systems, constant air volume) or by varying the cold and hot airs' airflows (VAV systems, variable air volume) both at a nearly constant or constant temperature rate. When a constant airflow rate is required the CAV systems are convenient. The advantage of CAV systems are that the fan always operates at maximum efficiency though this is not energy efficient. VAV systems are used where the thermal and airflow load requirements vary together, such cases are e. g. assembly halls, schools, etc., where occupants change with time and are the main heat load and pollutant source. In VAV systems, the variable airflow rate should be controlled by changing the fan speed by using variable frequency controller. Figure 2 and Figure 3 show the schematics of the CAV and VAV systems. [1]

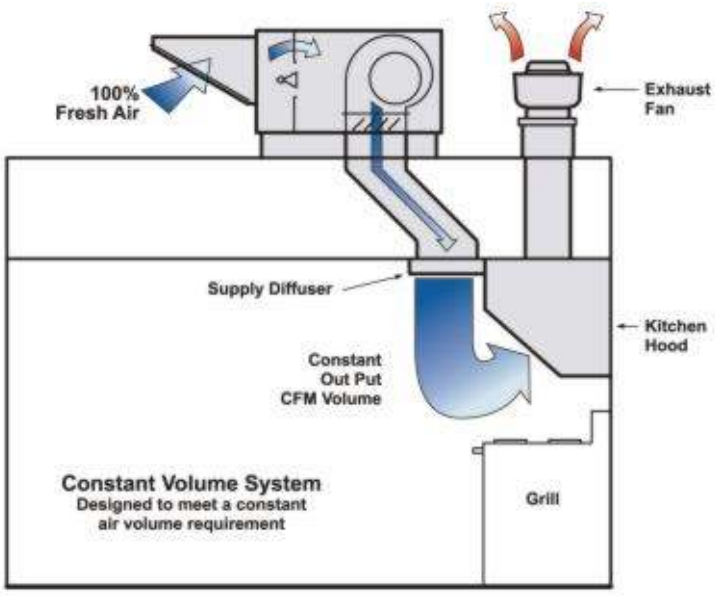


Figure 2. The schematic of the CAV system. [2]

2.1.3 Single-Duct Systems

In single-duct systems, at first the air is conditioned in an air-handling unit (AHU) to the required temperature and humidity. During the cold periods, the air should be colder and dryer than the indoor air, and vice versa during the warm period. The air is further distributed via a single network. [1]

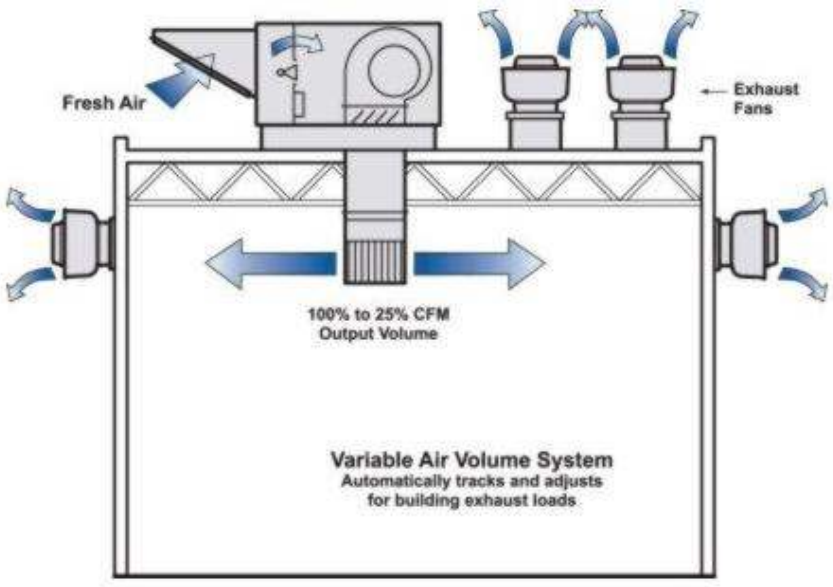


Figure 3. The schematic of the VAV system. [2]

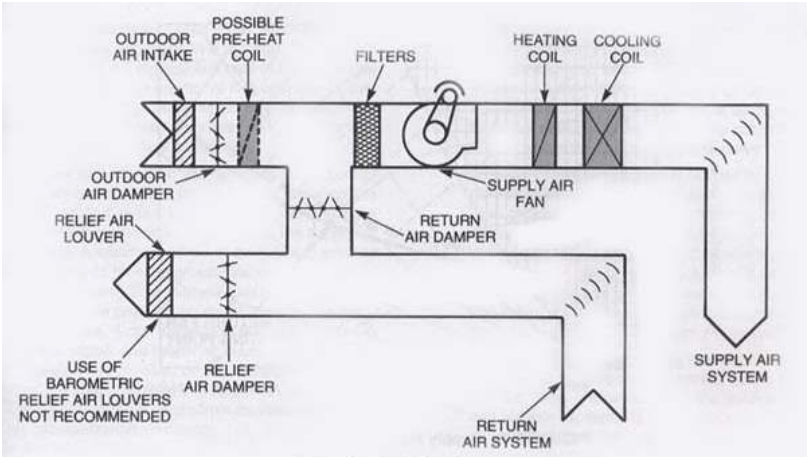


Figure 4. The schematic of the single-duct system. [3]

In CAV systems, AHU is used to control the temperature and humidity to maintain the required room climate when the load changes. In VAV systems, in order to cope with the room loads the fan speed or dampers are adjusted so that there is just enough supply airflow rate. Figure 4 shows the schematic of a single-duct system. [1]

2.1.4 Dual-Duct Systems

In dual-duct systems, the supply air is partly cooled and partly heated separately. Further the cold and hot air are distributed in two separate duct networks to the conditioned spaces and mixed in the proper proportions close to each ventilated spaces in order to compensate for the space load. These systems can also control the room climate either by changing the supply air temperature and humidity or by providing more or less supply air at a given temperature and humidity. Figure 5 shows the schematic of the dual-duct system. [1]

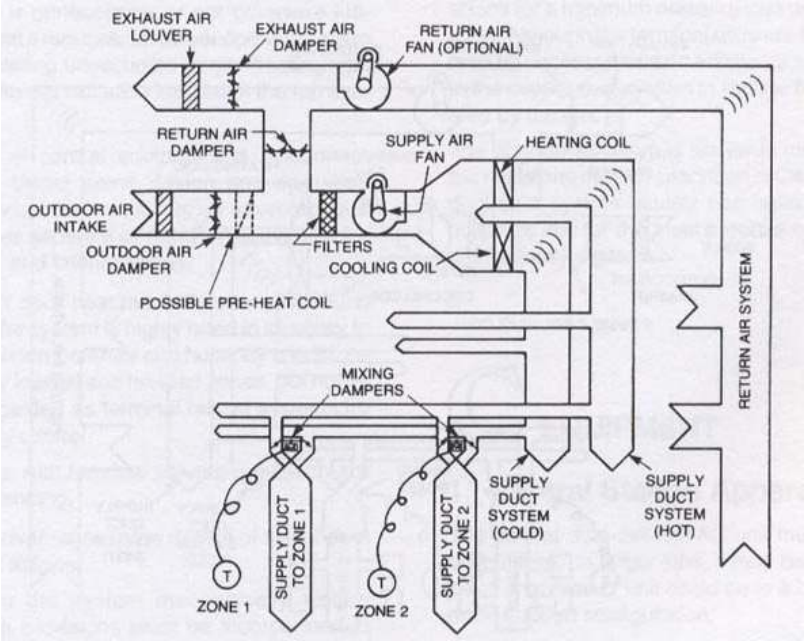


Figure 5. The schematic of the dual-duct system. [3]

2.2 Components of Air-Handling Units

Figure 6 shows schematically a supply and exhaust AHU. The outdoor air enters the unit through a louver and passes through a filter. Occasionally, when there is recirculation, the entered outdoor air is mixed with returning indoor air and the mixed air passes to a finer filter. Further the heat recovery system (if any) either pre-heat or pre-cool the mixed air and the air passes through the cooling and heating coils. Before the air is supplied to the air distribution ducts, a humidifier may increase the air humidity. After passing through the heat recovery unit (if any), the extracted air from the ventilated spaces is exhausted. Supply- or exhaust-only systems do not have heat recovery because they only use one half of this scheme. The main components of air-handling units are dampers, filters, fans, heating and cooling coils, humidifier, heat recovery and ductwork. [1]

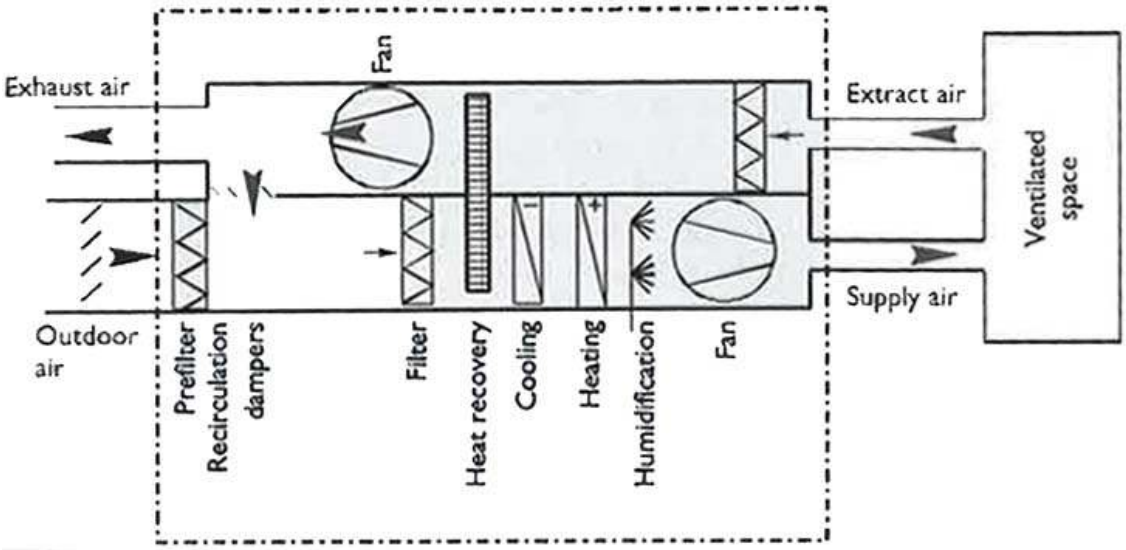


Figure 6. Components of air-handling units. [1]

3 AIR CONDITIONER ADJUSTMENT CONTROL METHODS

A control system is a device or interconnection of components forming a system configuration to adjust, regulate, command or direct the manner of other devices or systems. Control systems are developed to get the best out of a device or a system. Air conditioners are also adjusted using control systems to get high-performance outcome. The basis of the control systems are based on the linear system theory. The linear system theory implies a cause-effect relationship for the system components. The basis of the control systems is shown in Figure 7. [4]

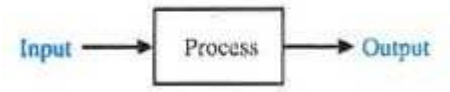


Figure 7. Basic idea of the control system. [4]

3.1 Control System Design

The control system design is a specific engineering design. The goal of the design is to achieve the configuration, specifications and identification of the key parameters of a system in order to meet the actual need. The control system design is illustrated in Figure 8. [4]

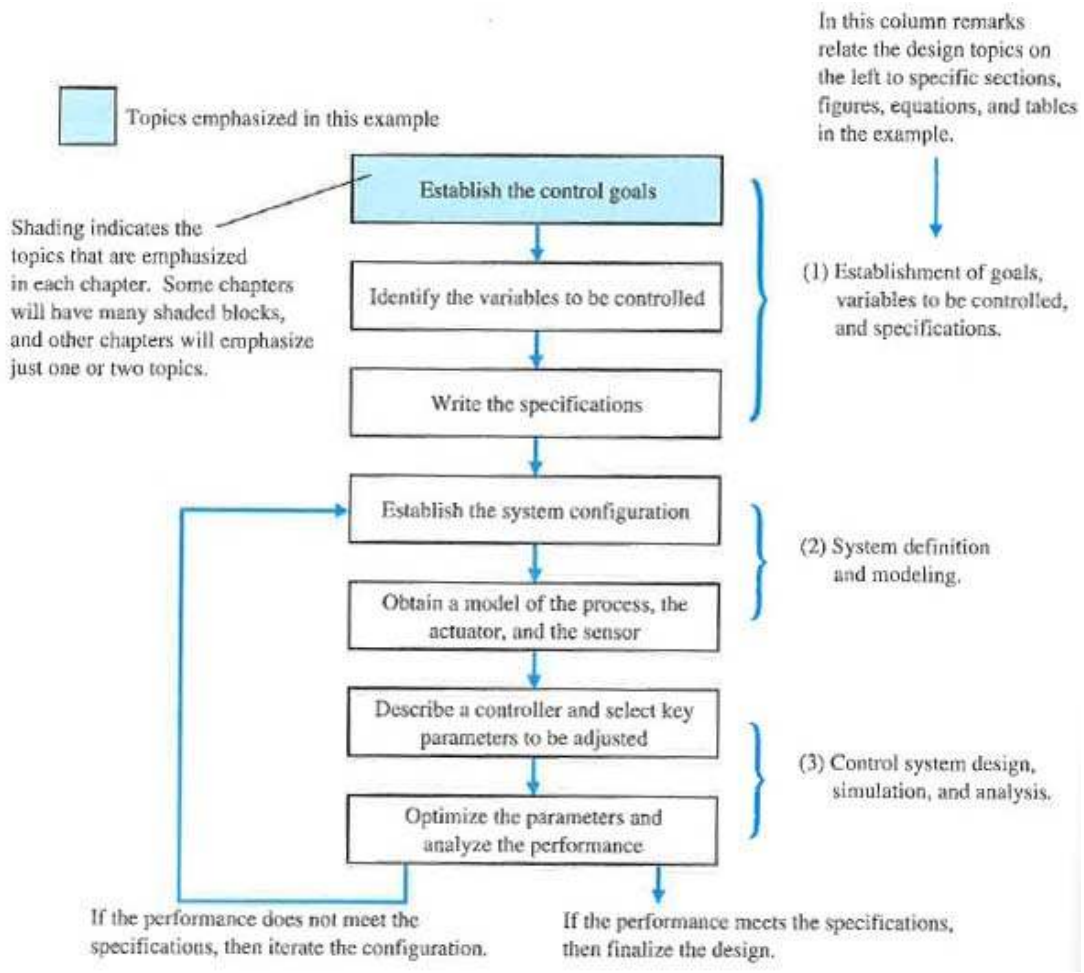


Figure 8. Control system design. [4]

3.2 Types of Control System

There are generally three different types of control systems, open-loop, closed-loop and multivariable. The input-output relationship of control systems illustrates the cause-effect relationship of the process shown in Figure 7. In the process the input signal provides an output variable, generally by the aid of the power amplification. [4]

3.2.1 Open-Loop Control Systems

Open-loop control systems use a controller and actuator to gain the wanted response, as shown in Figure 9. Open-loop control systems do not use feedback, so it means that the systems do not observe the output of the controlled process. Therefore, open-loop systems can not be involved in machine learning or either can correct any error cause by themselves. On the other hand open-loop control systems are beneficial for well-defined systems where the relationship between the input and output can be defined by mathematical formulas. An example of open-loop control systems is e. g. a microwave oven which is set to operate for a fixed time. [4]

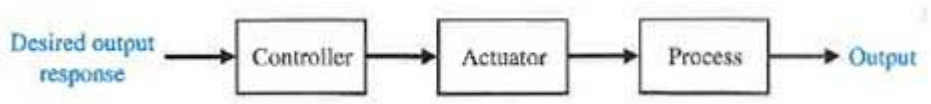


Figure 9. Open-loop control system. [4]

3.2.2 Closed-Loop Control Systems

Closed-loop control systems utilize an additional measure of the actual output in order to compare the actual output with the desired output response. The additional output measurement is called the feedback signal. A simple closed-loop feedback control system is shown in Figure 10. The system shown in Figure 10 is a negative closed-loop control system. Feedback control systems usually use a prescribed relationship function between the output and the reference input in order to control the process. The difference between the actual output and the desired output is equal to the error. The error is adjusted by a controller. The control system analysis and design use the feedback concept as their foundation. An example of closed-loop control systems is e. g. a person steering a car (assuming his or her eyes are open) by looking at the car's position on the road and making the convenient adjustment in order to steer it right.

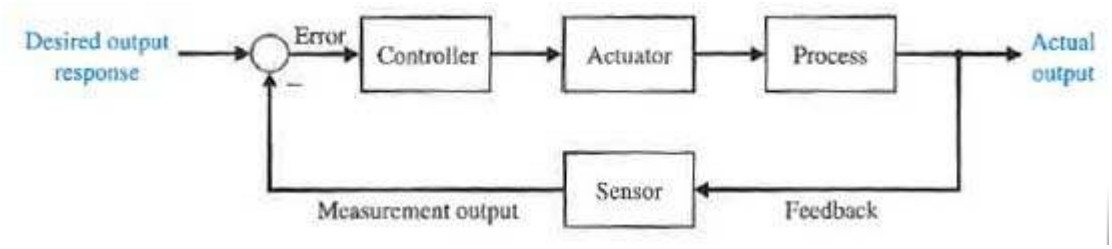


Figure 10. Closed loop feedback control system. [4]

3.2.3 Multivariable Control Systems

Nowadays control systems' complexity and interest in achieving optimum performance are increasing. The importance of control systems' engineering has been increasing since the past decade. Furthermore, the systems are becoming more complex and the interrelationships between several controlled variables are needed to be considered in the control scheme. Multivariable control systems solve these complex specification issues. Multivariable control systems consider feedback control systems with multiple inputs and multiple outputs and means systematically addressing modeling, uncertainty and performance in control systems. A block diagram representing a multivariable control system is shown in Figure 11. [4, 5]

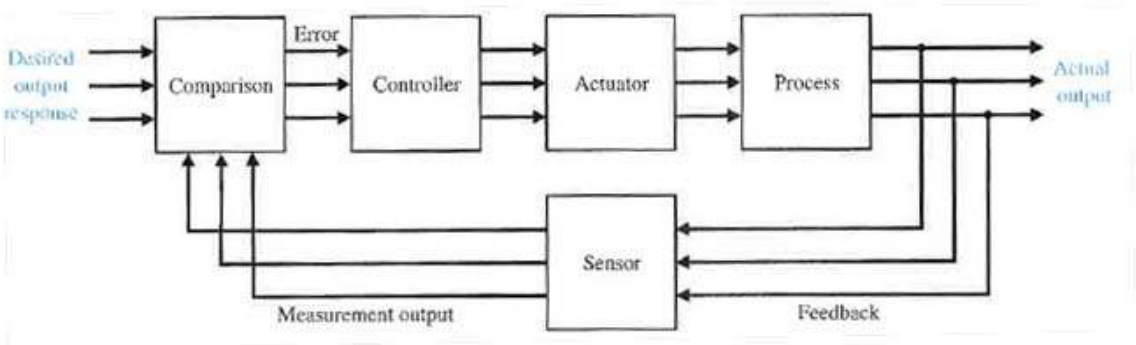


Figure 11. Multivariable control system. [4]

4 DESCRIPTION OF THE DEVELOPMENT TOOLS

A few obligatory development tools are needed to fulfill this project. In this chapter the obligatory tools and their functionalities are explained widely.

4.1 LabVIEW

LabVIEW is a graphical programming environment. The purpose of such programming is to develop sophisticated measurement, test, and control systems using intuitive graphical icons and wires that resemble a flowchart. LabVIEW programming automates the processes and measurement equipments used in any laboratory setup. The LabVIEW graphical language is also referred to as the G programming language. It was introduced in 1986 by NI (National Instruments) and since then LabVIEW has made its place in the top of the industrial world. LabVIEW is used by millions of engineers and scientists nowadays. It offers great integration with thousands of hardware devices and provides hundreds of built-in libraries for advanced analysis and data visualization. [7, 8]

4.1.1 Programming in LabVIEW

The G programming language is a dataflow programming language. The execution of this programming language is determined by the structure of a graphical block diagram (the LV (LabVIEW) source code). On the block diagram it is possible to place many different types of functions, also known as function-nodes. The programmer chooses suitable functions and connects them by drawing wires. These wires contain variables and as soon as a function gets all its required inputs, it is ready to be executed. The execution order can be defined by e. g. sequence numbering. The G programming language is capable of parallel function execution. An example of a typical LabVIEW VI block diagram is shown in Figure 12. [7, 8]

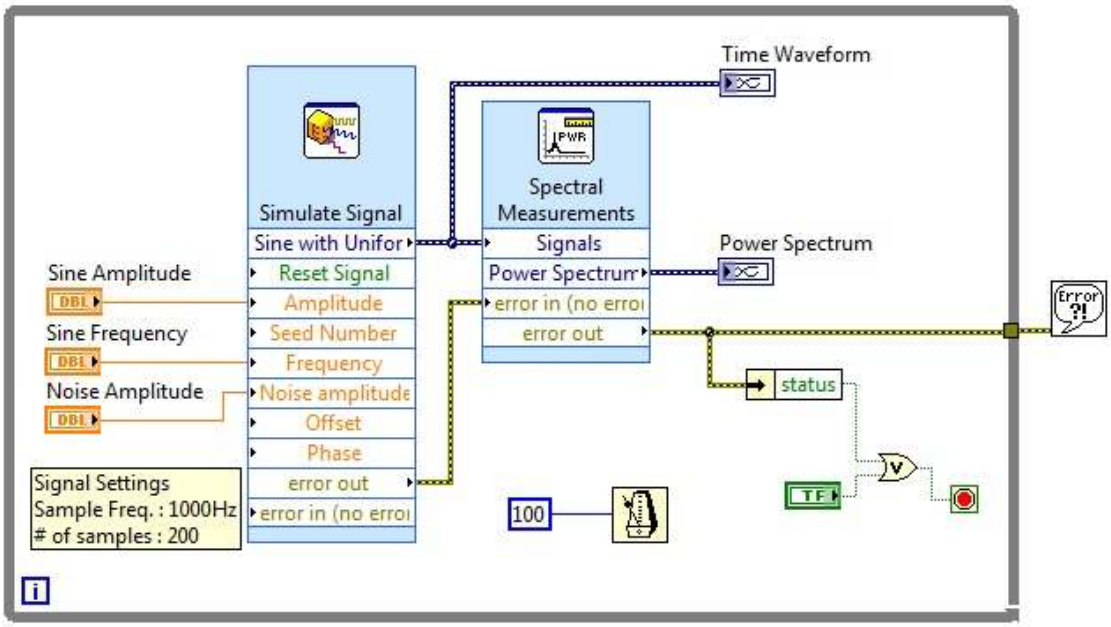


Figure 12. Typical LabVIEW block diagram.

LabVIEW programs are called as VI (virtual instrument). Each VI has three components: a block diagram, a front panel and a connector pane. The connector pane is used to represent a VI as a sub VI. The front panel is the only visual part for the user. The user can input data into or extract data from the running VI by controls and indicators located on the front panel. The front panel can also serve as a programmatic interface. Thus a LabVIEW program can be run as an individual program or as a sub program in any calling program. When using the VI as a sub VI function the inputs and the outputs of the function are defined from the front panel of the VI, using the connector pane. So, each VI can be tested before being embedded as a sub VI into a calling VI. An example of a LabVIEW front panel is shown in Figure 13. This is the front panel of the block diagram shown in Figure 12. [8]

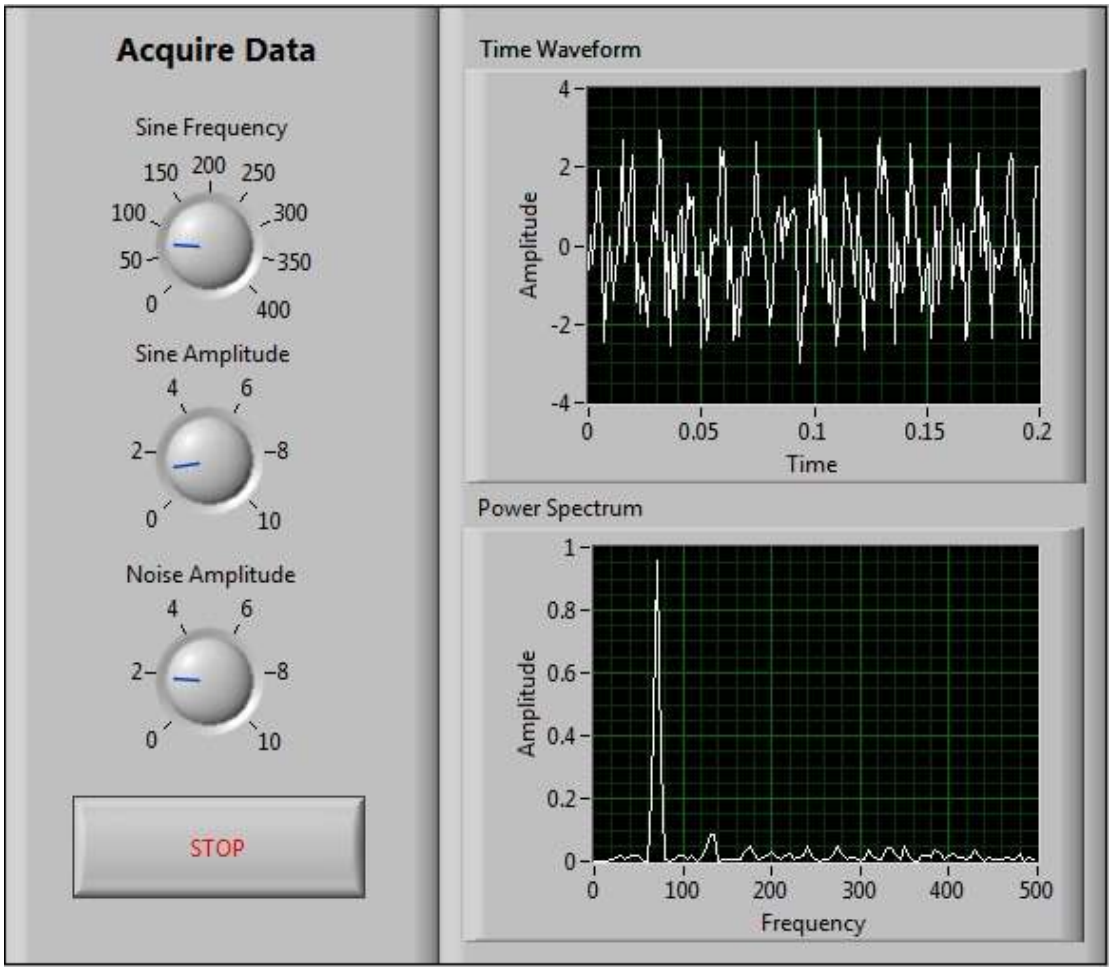


Figure 13. Typical LabVIEW front panel.

Being a graphical programming language LabVIEW allows people with a little programming knowledge and even non-programmers to create programs by dragging and dropping virtual representations (functions) available on the block diagram, which they are already familiar with. In LabVIEW programming environment with the help of available examples and documentations it's easy to create small applications. Though for creating larger applications it is essential that the programmer have an extensive knowledge of the special LabVIEW syntax and the memory management topology. [8]

The latest versions of LabVIEW software make it possible to build stand-alone applications. Stand-alone applications run individually and furthermore, it is possible to create distributed applications. Distributed applications communicate by a client or a

server scheme, thus they are easier to implement due to the inherently parallel nature of the G programming language. [8]

4.1.2 LabVIEW Workbenches

There are different kinds of ready structures in the LabVIEW programming language e. g. while loop, for loop, case structure, flat sequence, event structure etc. These structures are easily utilizable as a workbench for the LabVIEW source code. Using these structures as a workbench makes the coding simple and downright. In this chapter the structures used to complete the prototype are explained.

4.1.2.1 While Loop

A while loop repeats the subdiagrams inside it until the conditional terminal on the right down corner receives a terminating Boolean value. On the left down corner there is the loop iteration. The Boolean value depends on the while loop's continuation behavior. The conditional terminal can be designated to either stop or continue if the value is true. Also adding an error cluster to the conditional terminal and designate the continuation of the loop based on it is possible. Figure 14 shows a LabVIEW while loop. [9]

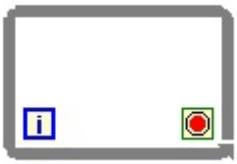


Figure 14. Labview while loop. [9]

4.1.2.2 Case Structure

Case structure has one or multiple stacked subdiagrams or cases. During execution only one of the subdiagrams or cases executes based on the value wired to the selector terminal. The value can be boolean, string, integer or enumerated type and determines which case to execute. Figure 15 shows a LabVIEW case structure. [9]

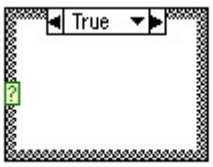


Figure 15. LabVIEW case structure. [9]

4.1.2.3 Flat Sequence

A flat sequence is sequentially executed frame consisting of one or more subdiagrams. A flat sequence is used to ensure that the subdiagrams are executed in a specific order and also to clarify the operation. The flat sequence structure has a different type of data flow than the other structures. The flat sequence frames execute from the left to the right and require all the data that are wired to the frame to be available in order to start executing. The data leaves each frame after executing it so the input of the following frame depends on the output of the previous one. Figure 16 shows a LabVIEW flat sequence structure. [9]

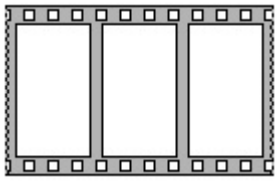


Figure 16. LabVIEW flat sequence structure. [9]

4.2 Step Motors and Step Motor Types

Step motors, also known as stepper motors are brushless or brushed synchronous electric motors that can divide a full rotation into a large number of steps. Step motors are electromechanical device which convert electrical pulses into discrete mechanical movements. When electrical command pulses are applied to the shaft of the step motors in the proper sequence, the rotation of the shaft increments in discrete steps. The motor rotation is usually directly related to the applied input pulses and the sequence of the applied pulses is directly related to the direction of the rotation. The rotation speed is directly related to the frequency of the applied input pulses and the rotation length is directly related to the number of pulses applied. There are three basic types of step motors, variable reluctance (VR), permanent magnet (PM) and hybrid (HB). The permanent magnet and the hybrid are the two most commonly used types of step motors. However, if uncertain about the right type of step motor for an application, evaluating the permanent magnet type is worth to try because of its low cost. There are also some special types of step motors e. g. the disc magnet motor. [10]

4.2.1 Variable Reluctance Step Motors

Variable reluctance step motors have been on the market for a long time. The functionality of variable reluctance step motors is probably the easiest to understand. Variable reluctance step motors consist of a soft iron multi-toothed rotor and a wound stator. In order to get the motor running the poles of the stator are magnetized by feeding DC voltage to the stator windings. As soon as the rotor teeth are attracted to the magnetized stator poles the rotation occurs. Figure 17 shows a cross section of a typical variable reluctance step motor. [10]

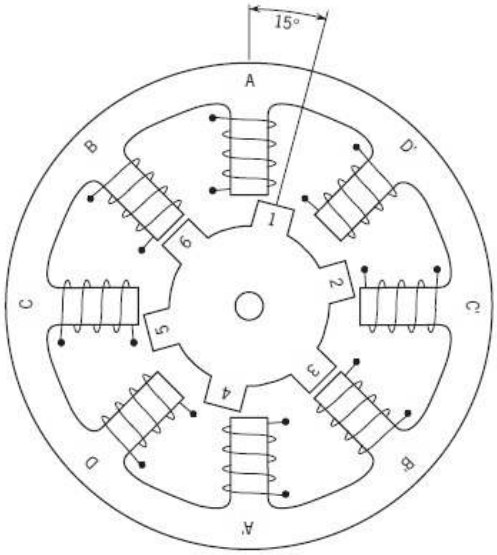


Figure 17. A variable reluctance step motor. [10]

4.2.2 Permanent Magnet Step Motors

Permanent magnet step motors are also referred as “tin-can” or “canstock”. Permanent magnet step motors are low cost and low resolution motors and they have a step angle range of 7.5° - 15° (48 – 24 steps per revolution). [10]

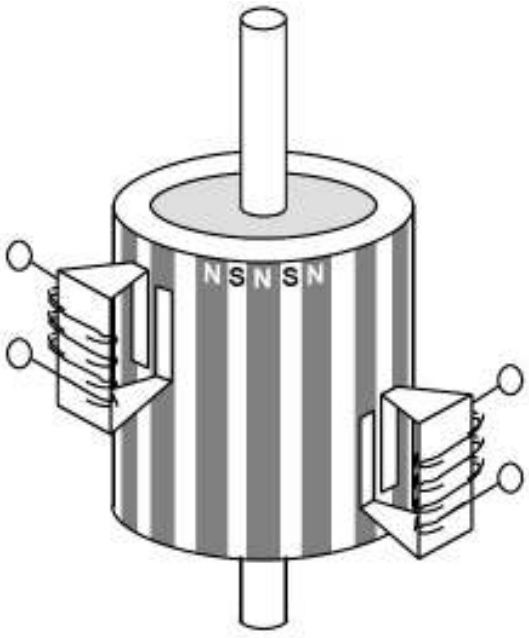


Figure 18. A permanent magnet step motor. [10]

As the name says permanent step motors have permanent magnets assembled in their internal structure. The rotor is magnetized with the alternating north and south poles situated in a straight line parallel to the rotor shaft. As the motors' poles become magnetized, it results into increasing the magnetic flux intensity which improves the torque characteristics. Permanent magnet step motors have better torque characteristics than the variable reluctance step motors. Figure 18 shows the principal of a permanent magnet or tin-can step motor. [10]

4.2.3 Hybrid Step Motor

Hybrid step motors are a combination of the best features of both the variable reluctance and permanent magnet step motors. Hybrid step motors are more expensive than the permanent magnet step motors but they provides better step resolution, torque and speed performance. Hybrid step motors have a step angle range of 3.6° - 0.9° (100 - 400 steps per revolution). The rotor of the hybrid step motors is multi-toothed like the variable reluctance step motors and their rotor shaft contains an axially magnetized concentric magnet around them. The rotor teeth improve the movement of the magnetic flux to the desired location in the air gap. Figure 19 shows the cross section of hybrid step motor. [10]

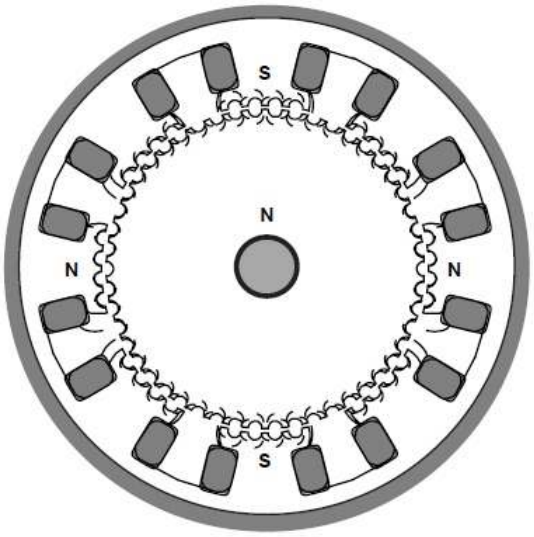


Figure 19. A hybrid step motor. [10]

4.2.4 Disc Magnet Step Motor

Disc magnet step motors have a disc shaped rotor with rare earth magnets. Figure 20 shows the principle of a disc magnet motor. These types of motors have a very low inertia and a non-coupling optimized magnetic flow between the stator windings. In some applications these types of step motors are the only appropriate type. [10]

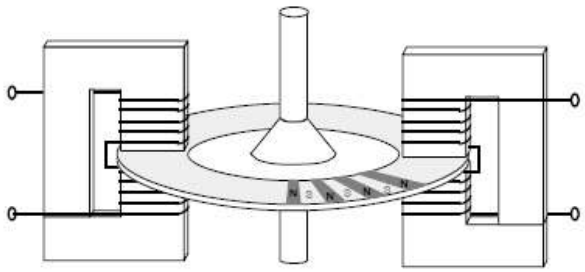


Figure 20. A disc magnet step motor. [10]

5 PROTOTYPE

The device is a prototype and planned to be developed further. There is a need to develop this kind of device in order to improve the air conditioning system of a garage. Particularly in winter, it is important that the air conditioner works at its maximum power while the snow melts from the top of the car, because the melting increases air humidity. The drying process is more power efficient if the air conditioner has its own automatic adjustment system compared to the situation where the potentiometer has to be set manually each time. It is also time saving if the adjustment could be done automatically. An effective air conditioner system is not only energy saving but also prevents the car from rusting. In this chapter all the issues related to complete the prototype are explained in detail.

5.1 Prototype and Its Basic Functionality

This device is developed to improve the performance of the air conditioning system. The requirements are that the device has to be portable and easily installable on any air conditioner. The device also has to be user friendly so that defining different humidity and temperature limits for different air conditioners in different places will be easy. The motor control system was programmed on a computer in order to make the limit resetting possible and easy for the users. The surfaces of the air conditioners are normally powder painted steel so the device is planned to be attached on the air conditioner using magnets. The basic architecture of the device is shown in Figure 21.

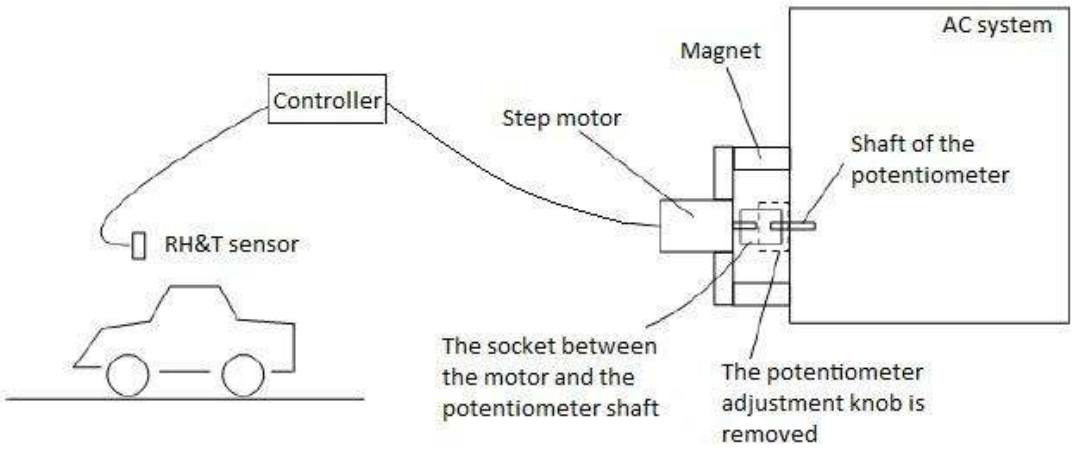


Figure 21. Basic architecture of the device.

The basic idea of the device is to adjust the air conditioner according to the air humidity and temperature readings. The values are read by using a device created using the RH&T (Relative Humidity and Temperature) sensor. The main program (Calling program) to run the device is done via LabVIEW to fulfill the adjustment control. Users can define the humidity and temperature limits when ever needed. The main program saves the latest readings of both humidity and temperature values in the memory and compares the memory values with the following readings in order to check if the defined limits for the humidity and temperature values are exceeded. If there has been any change since the last readings either in humidity or temperature values then the potentiometer is adjusted to its defined position based on the humidity or temperature limits.

5.1.1 Project Implementation

The implementation of the project consisted of the following five steps. At first the LabVIEW programming language was studied carefully and development planning was made to complete the device [6]. Secondly the reading device was connected to the measurement system and tested. A program was made via LabVIEW to read the data from it. Thirdly the technical and programming manuals for the step motor and step motor control was studied and the adjustment algorithm for the device was planned and

implemented [12, 14]. Fourthly, the main program was accomplished via LabVIEW by combining the reading device program with it. Finally, the device was consummated and was tested it in a real environment.

5.1.2 Components and Connections

The project is carried out by using an RH&T sensor, a PC, LabVIEW software, a step motor and a step motor controller. According to the requirement, the device is made using a step motor which automatically changes the power from outside without involving the electrical wiring of the air conditioner. The external connections are also kept simple. The reading device is connected to the PC's serial port RS232 and the step motor controller is connected to the PC's USB port. The reading device gets its needed power from the PC through the serial port. The step motor controller is connected to the step motor and also to a power source.

5.2 Hardware

The hardware needed for the project are a step motor, a step motor controller and a device to read the humidity and temperature values made using a RH&T sensor as already been mentioned. These hardware's characteristics and their functionalities are extensively explained below.

5.2.1 Step Motor

A step motor (Nanotec, ST4118X1404, NANOTEC ELECTRONIC GmbH & Co, Germany) is used for this project. ST4118 step motors provide high resolution with a reasonable price. These types of step motors are convenient for precision applications. They offer the highest possible torque and are also small in size (42 x 42 mm - Nema 17)**.** ST4118 step motors offer an excellent cost-effective alternative to the multiphase step motors in conjunction with the IMT and SMC drivers and with the same or even

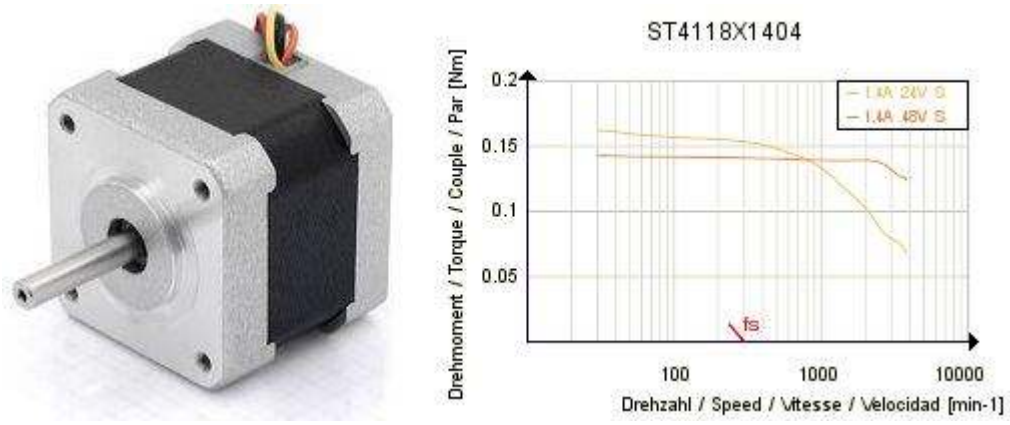


Figure 22. ST4118 step motor and ST4118X1404 step motor's torque curve. [11]

higher resolution of <0.36°. Figure 22 shows the step motor and its torque curve used for the project. The data sheet for the step motor is shown in Appendix 1. [11]

5.2.2 Step Motor Controller

The step motor controller (Nanotec, SMCI33, NANOTEC ELECTRONIC GmbH & Co, Germany) used for this project is a close loop positioning controller with encoder input. The step motor controller is shown in Figure 23. SCMI33 step motor controllers have compact microstep final output stage. They are flexible and easy to operate and also favourably priced. They have an adjustable microstep control of 200 to 25,600 steps per revolution which enables quiet, uniform running behavior, reduces system resonances and increases the resolution and the interchangeability to 3-phase and 5-phase motors at the same time. A phase current of 3 A phase and a voltage of 12-48 V enables to reach the speeds of more than 3000 rpm. SCMI33 step motor controllers are 90 x 60 x 22 mm in size and have both RS485 and USB interfaces. The system can then be started either in the automatic mode from the PC or in the standalone mode with an external signal. Alternatively, all commands can also be operated via the well documented ASCII code. Multiple axle applications are also possible via the smallest PLC. The data sheet of the SCMI33 step motor controllers is attached in Appendix 2. [12, 13]

Figure 23. SCMI33 step motor controller. [12]

5.2.3 Humidity and Temperature Sensor

The device used to measure the humidity and the temperature was made for another project in Savonia University of Applied Sciences, Information Technology R&D Unit. The sensor used in the device is a Sensirion SHT75 - Digital RH&T (Relative Humidity and Temperature) sensor and the sensor is shown in Figure 24. SHT75 digital humidity and temperature sensors are fully calibrated high-quality version of the pin-type humidity sensor series with cutting edge accuracy. They have a 80uW (at 12bit, 3V, 1 measurement per second) energy consumption, 0 – 100% RH humidity operating range, -40 – +125°C (-40 – +257°F) temperature operating range, 8 sec (tau63%) RH response time and a digital (2-wire interface) output. [15]

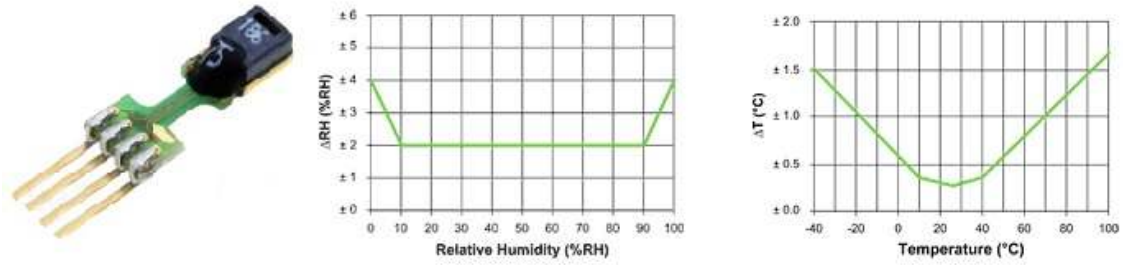


Figure 24. The sensor and its humidity and temperature curves. [15]

Beside the SHT75 sensor the device also contains a PIC (peripheral interface controller) microcontroller. The device is programmed in a way that it reads a 45-byte long string value every five seconds and sends the data through a bus to the microcontroller. The 45-byte value contains the reading time, the temperature value in both Celsius and Fahrenheit and the humidity value in percent. The microcontroller passes the data to the computer serial port RS232, from where the 45-byte value is later read. Figure 25 shows the reading device and its structure.

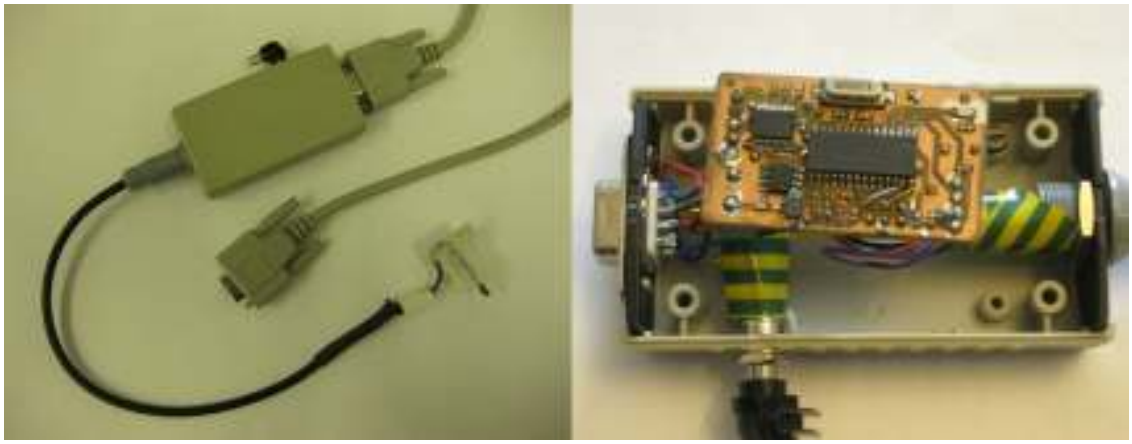


Figure 25. The Reading Device and its structure.

5.3 Software

The software part of the project is done entirely via LabVIEW. Two differrent LabVIEW programs are made and later combined to run the prototype. In this chapter the coding logics and functionalities of the programs are explained extensively.

5.3.1 Reading and Parsing the Sensor Value

A LabVIEW program is made to read the computer serial port and parse the wanted values out of the picked data. The temperature value in Celsius and the humidity value in percent are parsed from the 45-byte long string data. The limits defined for the temperature and humidity in the main program are also the same singular. Further this reading and parsing VI is embedded as a sub VI into the main program (calling VI). The front panel of the reading and parsing VI is shown in Figure 26.

In the VI's block diagram at first the input port name is defined so, that the program knows from where to read the data. The data is then wired to a property node function-node. The LabVIEW property node gets (reads) and/or sets (writes) properties of a reference [9]. This function can be used to get or set the properties and methods on local or remote application instances, VIs, objects and to access the private data of a LavbVIEW class [9]. In this program the property node returns the number of bytes currently available at the serial port [9]. A LabVIEW property node is shown in Figure 27. A delay is feed between the serial port reading and the property node to avoid unnecessary dataflow between them. The delay is located in a single framed flat sequence so that it always operates at the right time.

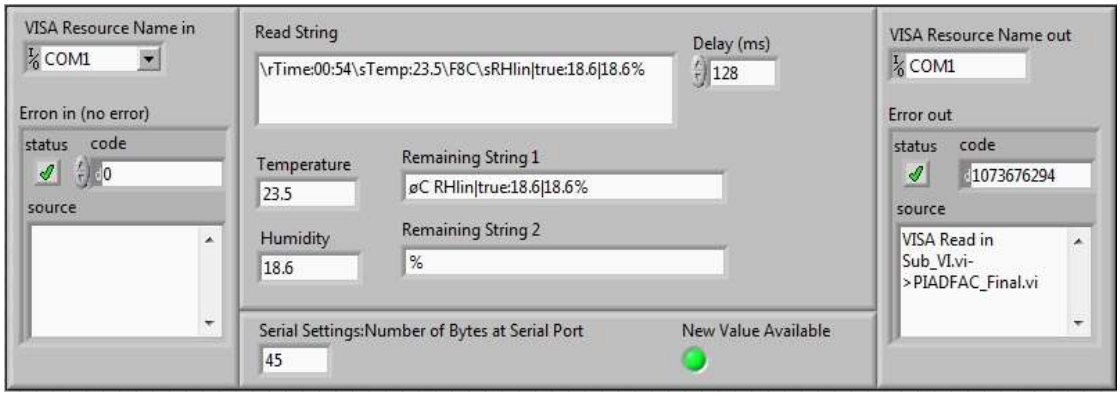


Figure 26. The front panel of the reading and parsing VI.

To make sure that the temperature and humidity values are available for reading from the sensor the block diagram code is designed in a way that the program executes only if the sensor picks a 45-byte long value. In order to achieve this functionality the reading value (byte amount at the port) is compared with number 45 to verify if they are equal. Further the comparison outcome is wired to the case selector of the reading and parsing VI, so if the outcome is true (the values are equal) the true case structure of reading and parsing VI executes. If the value is false (the values are not equal) the false case structure of reading and parsing VI executes. An error wire is carried through all the structures and functions in order to notify possible error. Figure 28 and Figure 29 show the true and the false case structures of the reading and parsing VI. The functionality of both of reading and parsing VI's case structures are explained below.

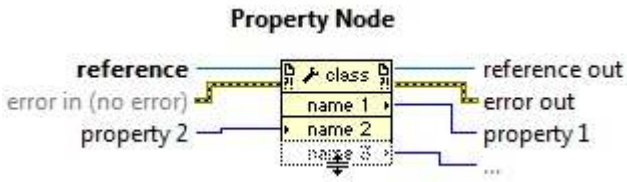


Figure 27. LabVIEW Property Node. [9]

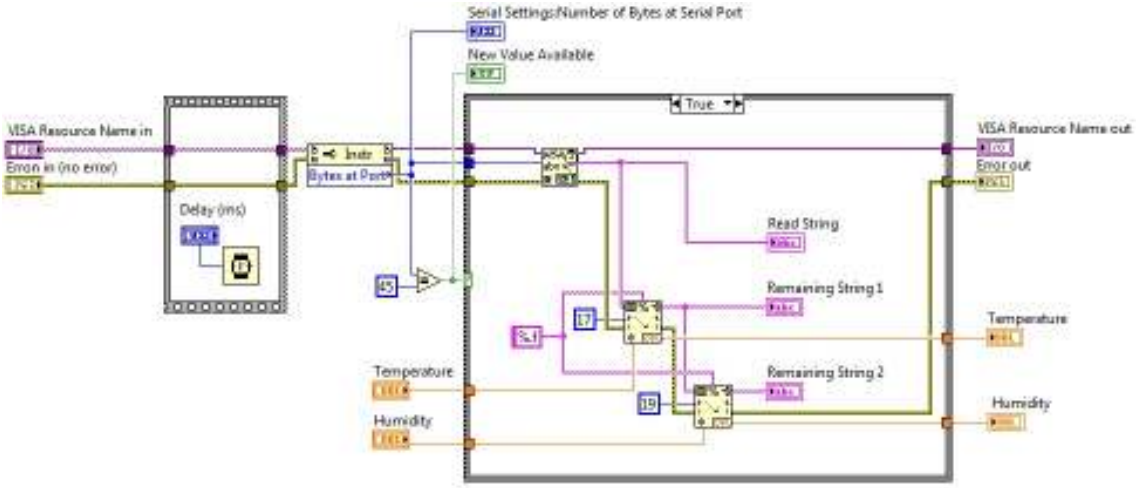


Figure 28. The true case structure block diagram of the reading and parsing VI.

In the reading and parsing VI's true case structure the data and the byte amount read from the serial port are wired to a VISA (visual instrument software architecture) read function as shown in Figure 28. The VISA read reads the specified numbers of bytes from the device specified by the VISA resource name input and returns the data in read buffer [9]. A LabVIEW VISA read function is shown in Figure 30.

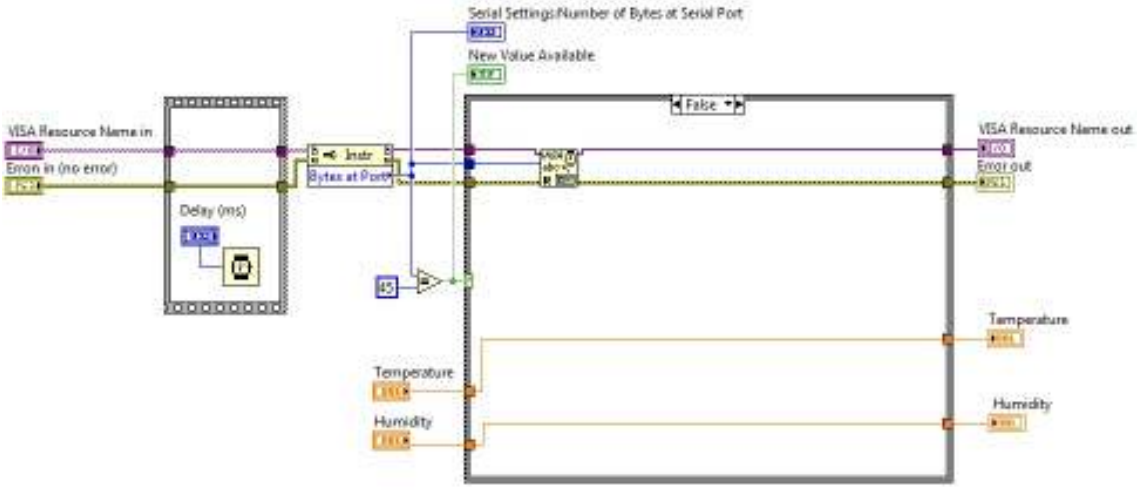


Figure 29. The false case structure block diagram of the reading and parsing VI.

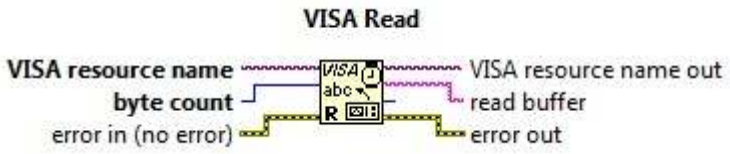


Figure 30. LabVIEW VISA read function-node. [9]

Further the read buffer value is wired to a scan from string function as shown in Figure 28. This function scans the input string and converts the string according to format string [9]. This function also can parse the string according to the byte count input [9]. A LabVIEW scan from string function is shown in Figure 31. The scan from string function is formatted to convert the string into floating point (%f) and then read the string from the 17th byte as shown in Figure 28 because the temperature value starts from there as shown in Figure 26. As the string is converted in floating point, the program reads until a non-floating type value come across. As soon as it scans a non-floating type value the scanning is terminated and the temperature and remaining string values are output as shown in Figure 26. The remaining string is then again wired to another scan from the string function in order to get the humidity value. The humidity value is also got in the same way as the temperature value, only this time the string is read from the 19th byte because in the remaining sting the wanted humidity value starts from there as shown in Figure 26. The byte places for both temperature and humidity values are got by simply counting from the string. In the reading and parsing VI's false case structure the data is feed to a VISA read function and passed through without any operation. In the front panel the read buffer (Read String) and both remaining string values are displayed by string indicators and the byte read, temperature and humidity values are displayed by numeric indicators.

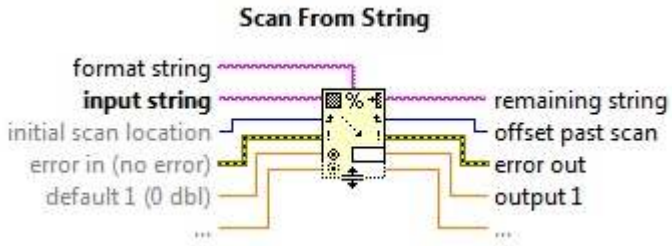


Figure 31. LabVIEW scan from string function-node. [9]

5.3.2 Main Program to Run the Device

The main program is the calling VI which calls for the reading and parsing sub VI. The aim is to keep both front panel and block diagram of the main program easily understandable and simple. The front panel and block diagram of the main program including their logics, functionalities and the significant function are extensively explained in this chapter.

5.3.2.1 Front Panel of the Calling VI

It is important to keep the front panel easily understandable because this is the only visible part by the users and the users utilize the device through this. The front panel of the main program is shown in Figure 32. The front panel is divided into two different parts, motor control and control limits. In the motor control part the session port definition, 1 step runtime for the potentiometer, the current position of the potentiometer, the current humidity and temperature values, the automation switch, the exit button and the history graph of the humidity and temperature values are located. By switching on or off the automation switch (Turn ON/OFF Automatic Control) users can set the air conditioner to run automatically or manually. The humidity and temperature limits are located in the control limits part. The session in is a combo box string singular, so it recognizes all the input ports in use and from there the right port can be chosen. The 1 step runtime value and the humidity and temperature limits are numeric control singulars so that the users can change the values easily when needed. The humidity and temperature limits are also expressed in numeric thermometer singulars for visual clarification though the limits can also be reset using them. The current position, the current temperature and humidity values are also represented in numeric singulars. The humidity and temperature as a function of time graph is a graph chart singular. The turn on/off automatic control and the exit program buttons are Boolean singulars.

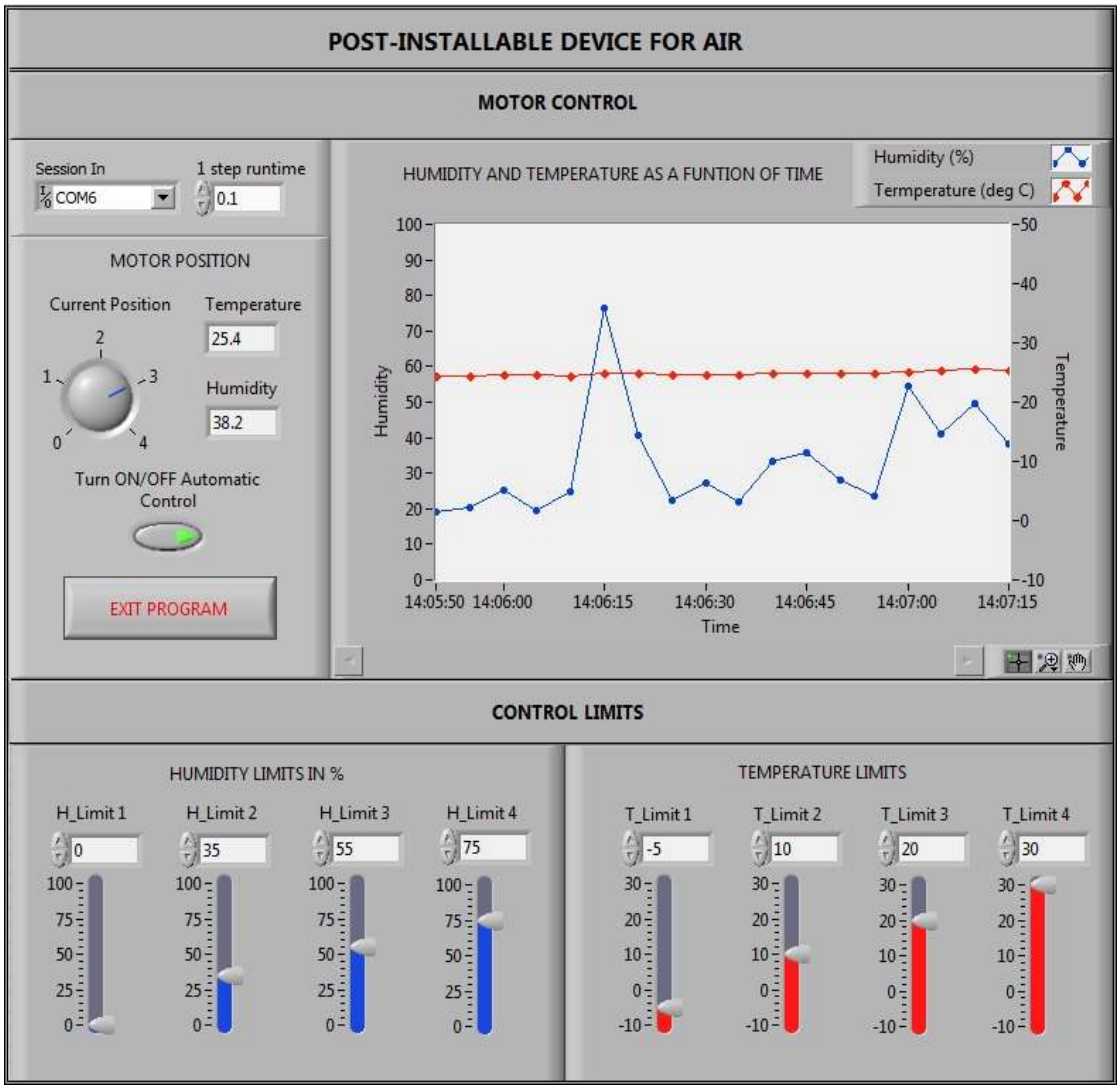


Figure 32. The front panel of the calling VI.

5.3.2.2 Block Diagram of the Calling VI

The whole source code is put in a double framed flat sequence structure as shown in Appendix 3. In the first frame the input source is defined and the motor set up is completed using an instrument I/O assistant function. This function is used to communicate with the message-based instruments and graphically parse the response e. g. communication between the instruments that use a serial, Ethernet or GPIB interfaces [9]. Figure 32 shows a LabVIEW instrument I/O assistant and its characteristics. The inputs and data types of the instrument I/O assistant are defined in order to get the right kind of output from the step motor as shown in Figure 34. Then the output data is passed forward.

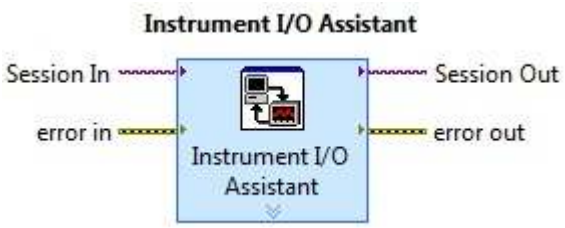


Figure 33. An instrument I/O assistant. [9]

An error wire is carried through the whole program and wired to the General Error Handler.vi in order to get possible error notification. If an error occurs this VI returns a description of the error and optionally displays a dialog box. Figure 35 shows the characteristics of General Error Handler.vi. A 50 milliseconds delay is put in the first frame to reduce unnecessary dataflow between the PC and the step motor controller.

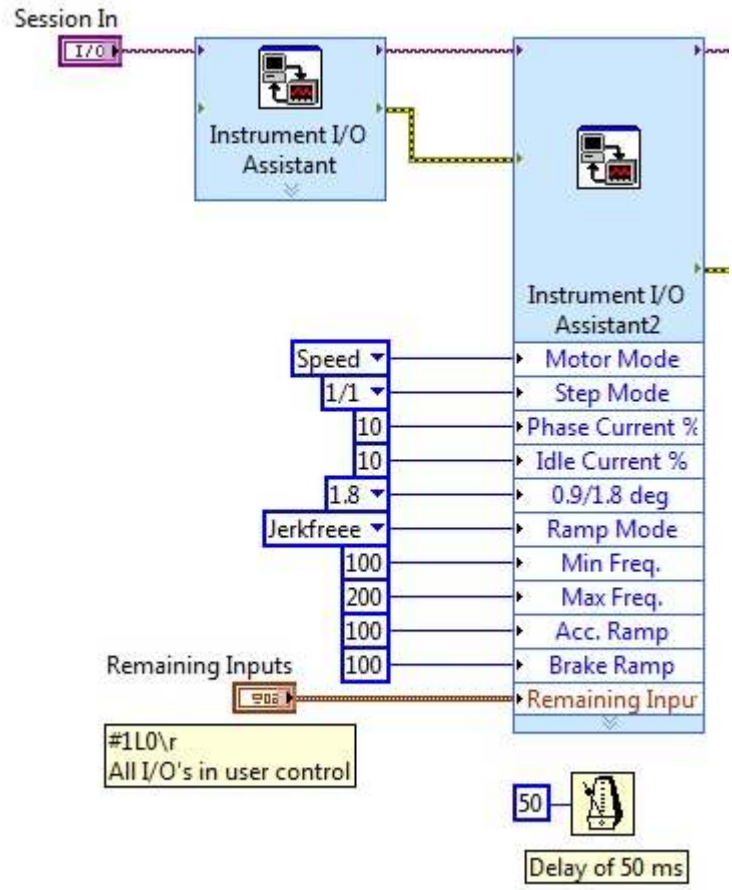


Figure 34. Motor setup of the main program.

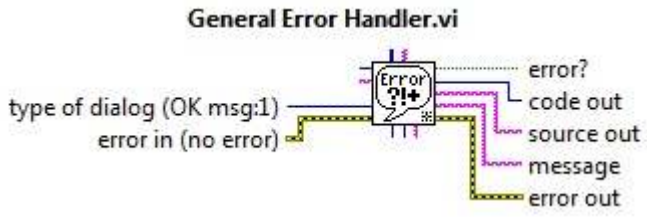


Figure 35. The General Error Handler.vi. [9]

In the second frame the inputs and data types of the reading device connected to the PC serial port are set upped. The history graph data acquiring and the control code are also located in the second frame. A VISA configure serial port function is used to read the serial port to which the reading device is connected to. The inputs and data types of it are defined in order to get the desired output as shown in Figure 36. The VISA configure serial port function is the tool to initializes serial ports in LabVIEW [9]. It outputs the picked data and possible error. Figure 37 shows a LabVIEW VISA configure serial port and its characteristics. The output of the VISA configure serial port is carried though each structures of the block diagrams and propelled through a VISA clear function. The VISA clear function clears the input and output buffer of the device [9]. A VISA clear function-node is shown in Figure 38. Further the output is wired to a VISA close function. The functionality of the VISA close function is to close the session to serial port [9]. Closing the serial port allows it to be used by other applications without quitting LabVIEW. Figure 39 shows a VISA close function.

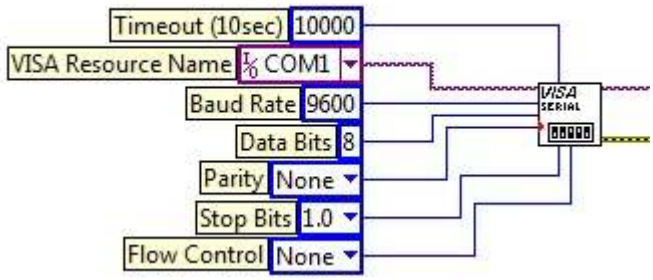


Figure 36. Modified VISA configure serial port.

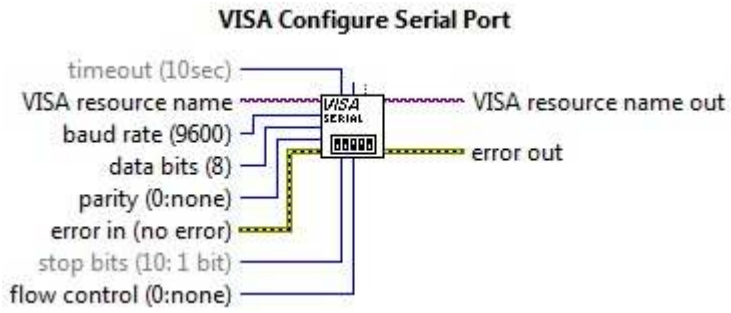


Figure 37. A LabVIEW VISA configure serial port. [9]

The history graph data are got by using a property node function. The functionality and characteristics of the property node is explained earlier in chapter 5.3.1. The current time for the history graph is achieved by using the get date/time in seconds function and by converting the output in seconds, minutes and hours by using the LV70TimeStampToTimeRec.vi sub VI. The output is then wired to the unbundled by name function in order to get the outputs of the specified cluster elements. Unbundle by name function returns the cluster elements whose names are specified [9]. The sub VI is available in LabVIEW function tools for such conversions. The get data/time in seconds function returns a timestamp of the current time [9]. LabVIEW calculates this number of seconds elapsed since 12:00 a.m., Friday, 1 January 1904, Universal Time [9]. The time format is then again converted into seconds using convenient numeric functions to get the suitable input for the XScale.Offset as shown in Figure 40. A constant 5 is wired to the XScale.Multiplier so that the graph marks the data every five seconds as the data is read every five seconds. The constants 0 and 4320 are wired to the XScalse.Minimun and XScale.Maximum to save the history in memory. 4321 is the maximum memory place for the XScale.Maximum and in this case it saves the history of one month in it. An initialize array function is used to save the history in memory. The initialize array function creates an n-dimensional array in which every element is initialized to the value of element [9].

Figure 38. VISA clear function. [9]

Figure 39. VISA close function. [9]

It is required that the program sets the potentiometer at position 0 if the user wants to, otherwise the potentiometer is always at positions between 1 to 4 based on the readings. The purpose of the first case structure of the main program is to fulfill the requirement mentioned above. A button (Turn ON/OFF Automatic Control) is located on the front panel and the block diagram is coded so that by only pressing the button on the front panel the motor is commanded to set the potentiometer at position 0. The button indicator on the block diagram is connected to the case selector of the turn on/off automatic control case structure as shown in Appendix 3 (1). The true case structure of the turn on/off automatic control executes if the button is pressed on (the case selector receives a true value) and it means that the automatic adjustment device is turned on and the program runs normally. The false case structure of the turn on/off automatic executes if the button is not pressed on (the case selector receives a false value) and it means that the potentiometer is set to position 0 and the program is terminated so the air conditioner can be adjusted manually.

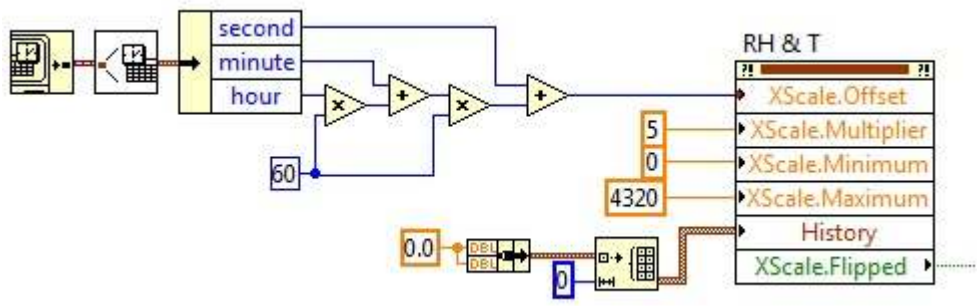


Figure 40. Current time and history graph code.

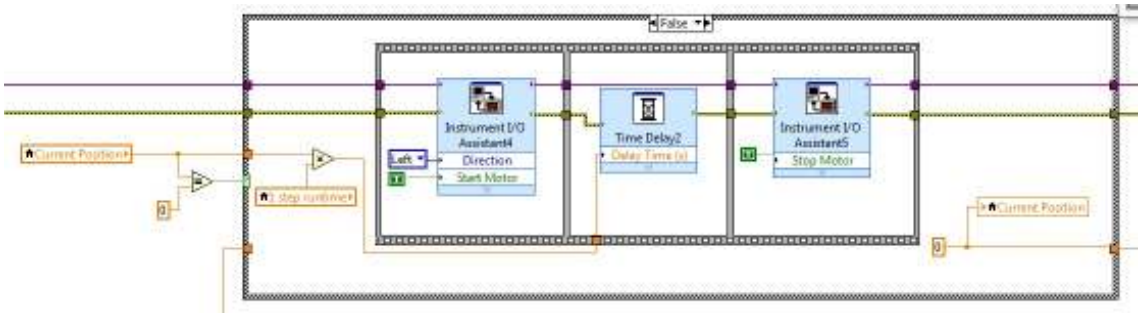


Figure 41. False case structure of set to 0.

To accomplish the desired functionality of the turn on/off automatic false case structure the current position of the potentiometer is compared with zero to check their equality. Zero represents the potentiometer position 0 as shown in Figure 41 and Figure 42. If the outcome is true (current position value is equal to 0) it means that the potentiometer is already at position 0. So, the program executes the true case structure of set to zero and the data just pass through without performing any operation as shown in Figure 42. But if the outcome is false (current position is not equal to 0) then the false case structure of set to zero executes to set the potentiometer at position 0.

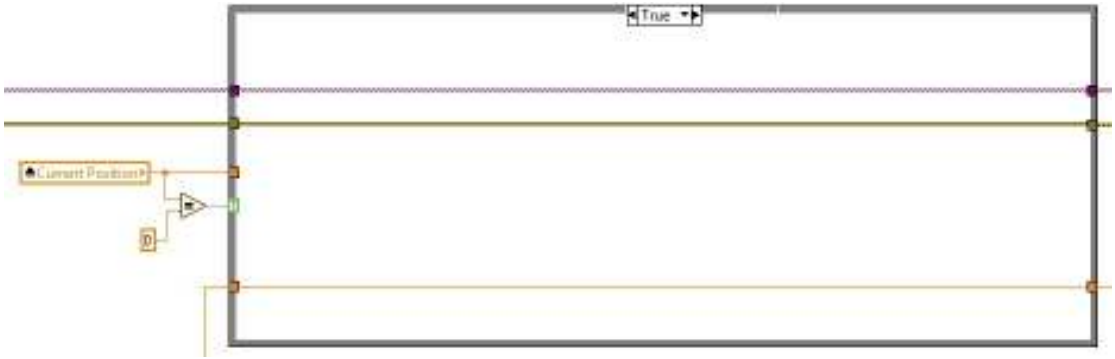


Figure 42. True case structure of set to 0.

In the set to zero false case structure execution the current position is multiplied with the runtime of 1 step of the potentiometer in order to determine the delay as shown in Figure 41. The delay is the runtime of the motor so it tells the motor how long to run. The 1 step runtime means how long the air conditioner potentiometer needs to reach from one position to the next, however, it is possible to move more than one step at once. Further the delay is feed to a time delay function as shown in Figure 41. The first instrument I/O assistant starts the motor and is defined to run only to the left in order to tell the motor run to the correct direction. A second instrument I/O assistant is used to input the motor termination command. Both instrument I/O assistants and the time delay function are located in a triple framed flat sequence structure in order to sequence and clarify the execution progression.

The limit comparisons and motor control are situated in the second case structure of the main program. The limit comparisons and motor control consist of altogether six case subdiagrams. Both the humidity and temperature values are compared in few different ways in order to accomplish the best and accurate motor controlling. The comparisons are explained later. In the first subdiagram (subdiagram -1) the humidity and

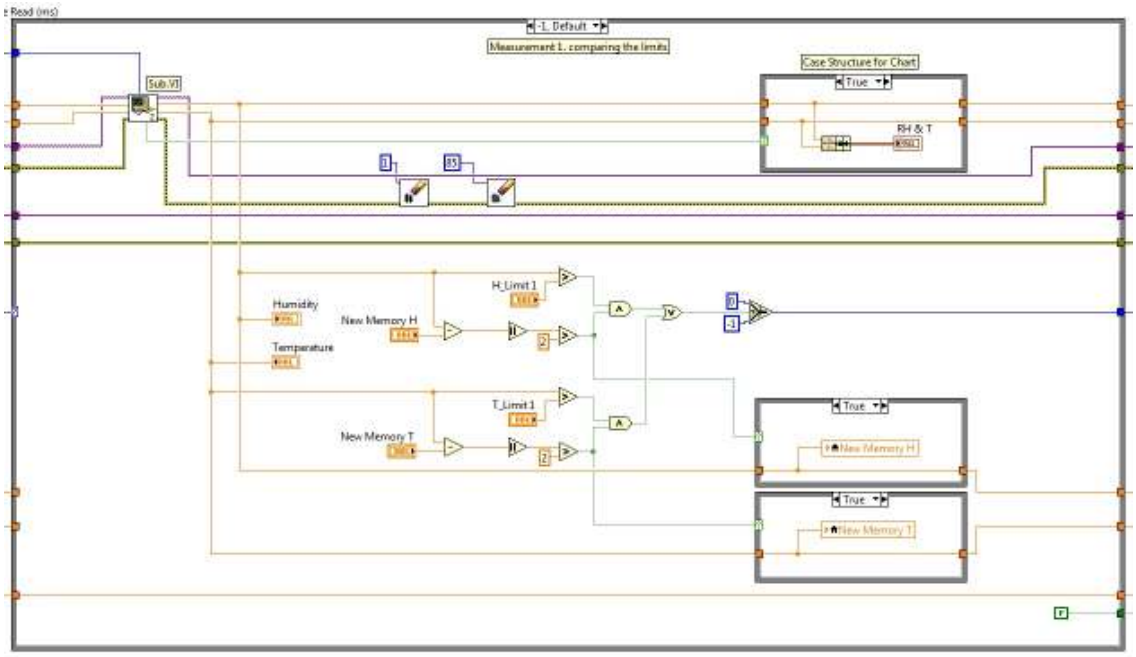


Figure 43. First limit comparison diagram.

temperature values achieved from the embedded reading and parsing VI are saved in the memory as New Memory H and New Memory T as shown in Figure 43. Both values are saved as local variables in order to easily use them in other subdiagrams. They are situated in case structures so only if there is any change in the picked values then the local variables' are replaced. The humidity and temperature values achieved from the reading and parsing VI are compared with the first limits defined by the user to check if the picked values exceeded the limits or not. In the next comparison the humidity and temperature values are subtracted by the New Memory H and New Memory T values. The subtraction outcome is propelled through an absolute value tool to always get positive values.

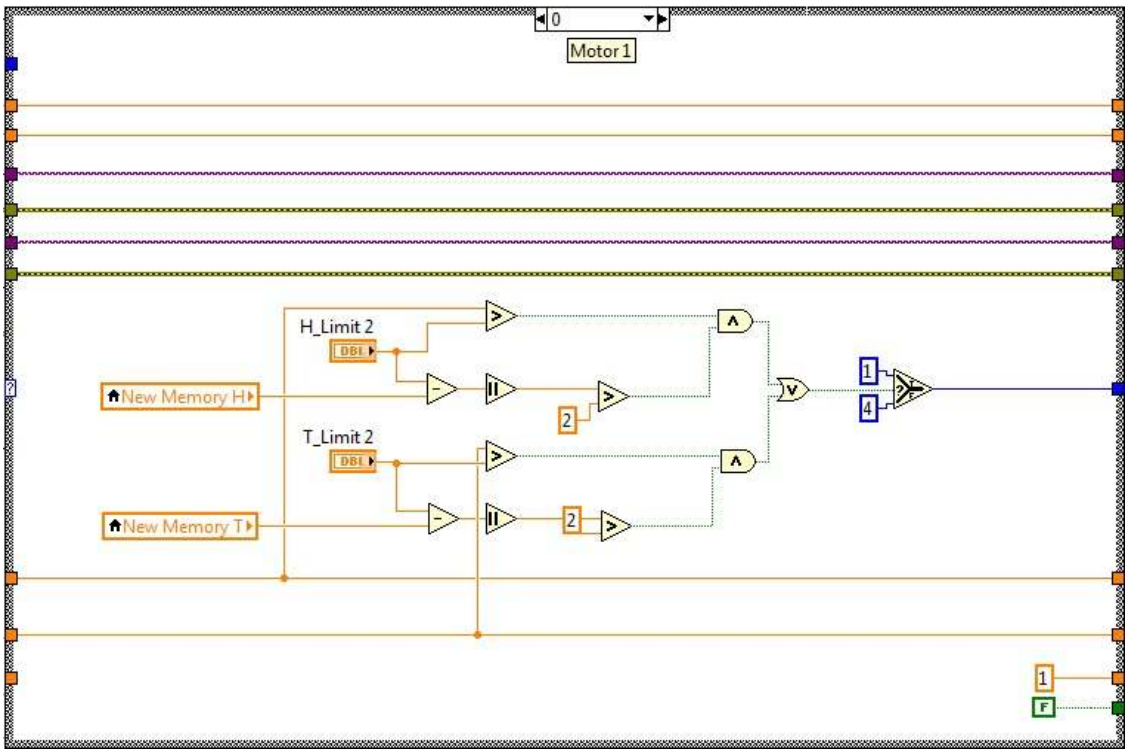


Figure 44. Second comparison diagram.

Further the outcomes are compared by two to find out if they are greater than that. If the outcomes are greater than two it means that there have been a change of at least two digits since the last readings in either humidity or temperature values and only in this case the program is passed forward for possible adjustments. This prevents unnecessary adjustments of the motor when the reading values are fluctuating near the limit values. If both limits' and subtractions' outcomes are true the execution is passed to the final comparison step where the outcomes of the previous comparisons are compared to verify if either of the outcomes is true. Further the final outcome is wired to a select function as shown in Figure 43. The select function returns the value wired to the upper input t or lower input f, depending on the value of s. If s is true, the function returns the value wired to t and if s is false, the function returns the value wired to f. If both of the outcomes are or are true the program determines the ruling factor according to the defined limits. Once the ruling factor is determined the program is passed to the next subdiagram (subdiagram 0). If none of the outcomes are true the program stays in this subdiagram (subdiagram -1) and goes through the whole operation all over again. The waveform chart is also located in a case structure in the first subdiagram as shown in Figure 43. The chart values are the temperature and humidity values achieved from the reading and parsing sub VI. The values are wired to a bundle function and the output of it further wired to the waveform chart input. Bundle function assembles cluster from individual elements [9]. The case selector of the waveform chart is wired with the new value available outcome of the reading and parsing sub VI function. So, the chart is marked each time a new value is available, otherwise nothing is marked in the chart.

In the following subdiagram the humidity and temperature values are compared with the second limits. Then the outcomes are passed to the select function and the ruling factor is determined exactly in the same way as in the previous subdiagram. The execution is passed to the next subdiagram (subdiagram 1) if the select function input is true. Otherwise, the program is sent to the motor control subdiagram (subdiagram 4) to set the potentiometer at position 1 as shown in Figure 44. The following two subdiagrams (subdiagrams 1 and 2) are also coded similarly as this one but by using the third and fourth limits. The program is then passed to the convenient subdiagram to set the convenient potentiometer position based on the outcomes as shown in Appendix 3.

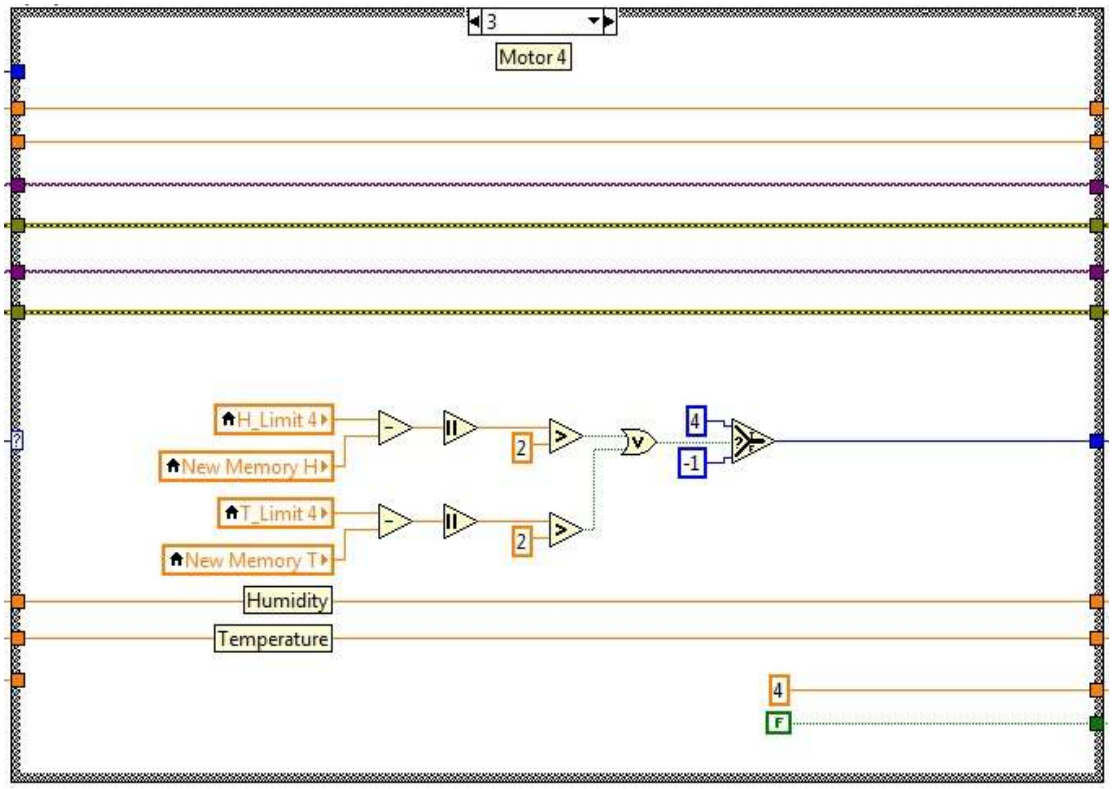


Figure 45. Fifth comparison diagram.

In the fifth subdiagram (subdiagram 3) both humidity and temperature memories are subtracted by their fourth limits and the outcomes are propelled through an absolute value tool and further subtracted by two to see if there is enough change in the values as shown in Figure 45. The ruling factor is discovered in the same way as in previous subdiagrams. The limit comparison is not needed here because this is the subdiagram of the highest limits. The outcome is then wired to a select function and if the wired input is true the program is sent to the motor control subdigram and the potentiometer is set at position 4. If the wired input is false the program is sent to the first subdigram (subdiagram -1) to execute the whole procedure all over again.

The sixth subdiagram (subdiagram 4) contains the motor control. The motor control is a closed-loop feedback control system. The code for controlling the motor is located in a case structure shown in Appendix 3 (6). The true case structure of the motor control executes if there is a change in the current memory value (Current Memory) and the measured

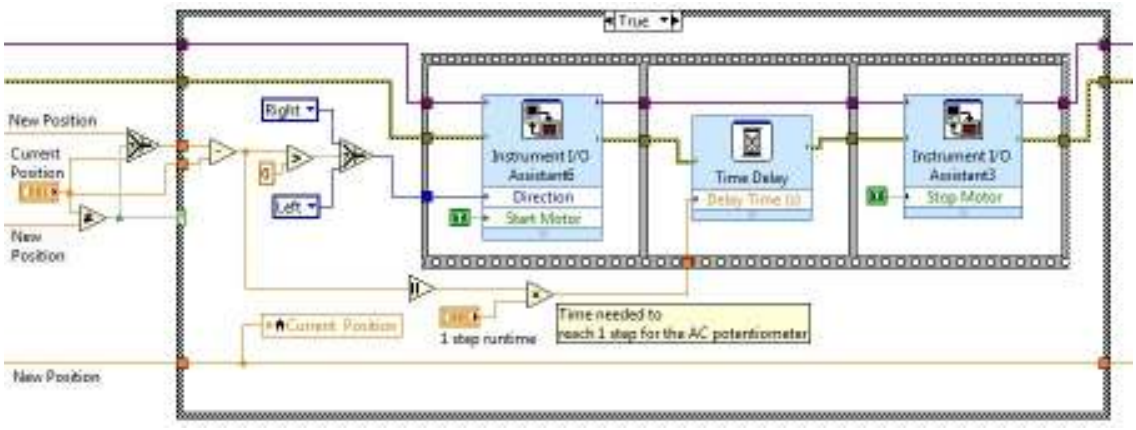


Figure 46. True case structure of the motor control diagram.

value (New Position). The current position and the new position are then compared in order to verify their non-equality. The outcome is then wired to the case selector of the motor control and to a select function as shown in Appendix 3 (6). If the outcome is true (there has been a change) the true case structure of the motor control executes and the select function outputs the new position value. If the outcome is false (there has been no change) the false case structure of the motor control executes and the data only pass through. Figure 46 and Figure 47 show the block diagrams of both true and false case structures of the motor control.

In the true case structure of the motor control the direction of the step motor rotation is defined by subtracting the new position and the current position values and comparing the outcome with zero to determine if the outcome is positive or negative. The positivity and negativity defines towards which direction the motor needs to run. Further the outcome is wired to a select function. If the outcome is true (outcome is positive) the select function outputs the upper input value (enumerated value 0, Right) and the step motor is set to run to the right. If the outcome is false (outcome is negative) it outputs the lower input value (enumerated value 1, Left) and the step motor is set to run to the left. Further, the outcome of the select function is wired to an instrument I/O assistant function. This instrument I/O assistant starts the motor by sending a start command and sets the direction according to the enumerated value.

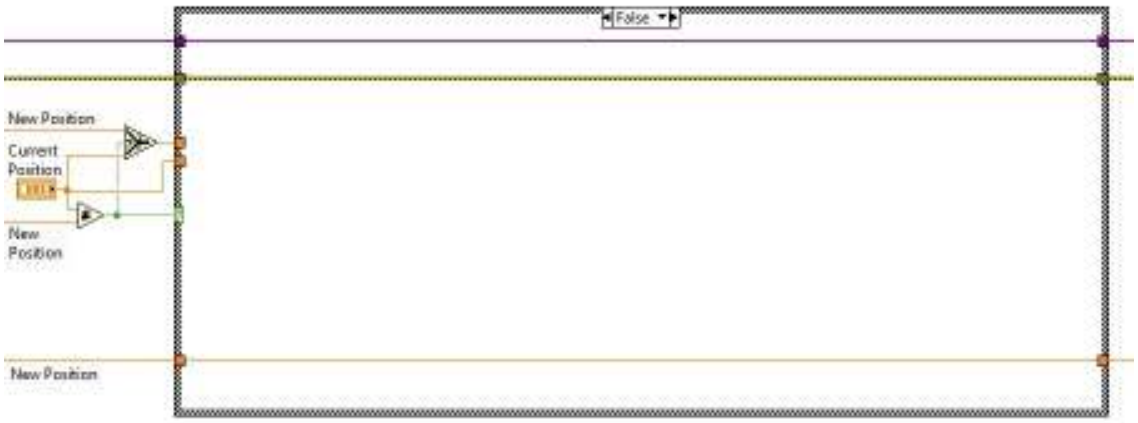


Figure 47. False case structure of motor control diagram.

The motor control true case structure is pretty similar to the set to zero false case structure. The motor runtime time is defined by feeding a delay to a time delay function. The delay is the multiplication of the current and new positions' subtraction reminder and the 1 step runtime. The subtraction reminder is propelled through an absolute value tool. The runtime is changeable from the front panel by the users. So, to use the device on different air conditioners the users only need to know the time needed to travel one step for the potentiometer of that specific air conditioner. The termination of the motor is defined by sending a stop command via another instrument I/O assistant.

6 DISCUSSION

In this chapter a rapid review of the project is given. The project is analyzed and the problems, benefits and changes considering this project are discussed. The project was completed without facing major problems. There was one prominent problem and luckily it was possible to solve without major modifications. The problem was in implementing the sub VI (reading the sensor) into the main program. Every time the main VI was being executed the buffer of the sub VI was over flowing and the whole execution was starting all over again. Few modifications in the sub VI and main code solved the problem and the program was running in the desired way.

During the working process few changes were made in the project. The original idea was to build the device only based on the humidity value, but at the processing stage also the temperature value was taken under consideration. Finally, the program was made so that the adjustment was done based on both the humidity and temperature values. Another change made was to leave out the idea of embedding the whole program into a microcontroller and instead do the whole programming only using the computer. The reason for the change was that embedding the LabVIEW program into a microcontroller would have made it difficult for the device to fit on different air conditioners. Because, once the program is embedded into a microcontroller, changing the humidity and temperature limits without interfering in the main code is not possible. Another reason for leaving this part out of the project was to keep the size of this final project suitable.

Nevertheless, this project was quite beneficial for me because I learnt many new things through this. I got a deep view of the functionality of step motors, RH&T sensors, air conditioning systems and control engineering. Most importantly I learnt the LabVIEW programming language.

During this project possible development ideas for the prototype was conceived. In order to make a commercial version of the prototype the computer should be replaced with a microcontroller. So, the main program of the prototype which is done via LabVIEW on a computer must be embedded into a microcontroller. Also power sources must be designed onto a circuit board. An encoder can be used to read the potentiometer positions to make the adjustment even easier. Using an encoder makes it possible to read the potentiometer position in angle and adjust the potentiometer based on that so the users do not need to find out the one step runtime. Further the circuit board can be designed so that the device will be capable of wireless data transferring e. g. through Bluetooth or USB modem.

7 SUMMARY

The project was a success, despite the problems the device works as it was planned to work. During the project possible product development ideas were conceived. In order to complete the project the LabVIEW programming language was studied at first. Then possible development plans to complete the device were made. A LabVIEW program was made to read and parse the sensor value read from the RH&T sensor. Further the step motor controller manual was studied to do the adjustments for the step motor. After that the main program was made via LabVIEW and the reading and parsing program was combined with it. Finally, the device was consummated and was tested it in a real environment. Learning LabVIEW was fun and learner friendly. It was important to get to know the basic components and their functionalities at the beginning. After gaining the basic knowledge creating basic applications was practiced. Thus the new components and their functionalities became familiar. LabVIEW is an awesome tool for such measurements and controlling.

REFERENCES

[1] B. A. Hazim, First published (2008). *Ventilation Systems, Design and performance. Taylor & Francis*.

[2] Rupp Air Management Systems, May 5, 2011 [online]. http://www.ruppams.com/catalogcontent/fans/sup_ram/types.asp

[3] Air Distribution Arrangements, May 5, 2011 [online]. http://www.hvacfun.com/f-air-distribution-arrangements.htm

[4] R. C. Dorf, R. H. Bishop, 11th Edition (2008). *Modern Control Systems.* Pearson Education, Inc.

[5] Multivariable Control, April 13, 2011 [online, PDF]. http://www.cats.rpi.edu/~wenj/ECSE646F06/lec01.pdf

[6] National Instruments, March 2004 Edition. *LabVIEW Basics II: Development Course Manual.*

[7] R. H. Bishop, (2010). *LabVIEW 2009, Student Edition.* Pearson Education, Inc.

[8] National Instruments, LabVIEW, March 14, 2011 [online]. http://www.ni.com/labview/

[9] National Instruments, *LabVIEW 2010, LabVIEW Help.*

[10] Industrial Circuits Application Note, Stepper Motor Basics, March 21, 2011 [online, PDF] http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf

[11] Nanotec, April 24, 2011 [online]. http://en.nanotec.com/steppermotor_st4118.html

[12] NANOTEC ELECTRONIC GmbH & Co. KG. *Technical Manual, Stepper driver SMC133*. Gewerbestraße 11 D-85652, Munich, Germany.

[13] Nanotec, April 14, 2011 [online]. http://en.nanotec.com/steppermotorcontroller_smci33.html

[14] *Programming manual for stepper motor positioning controls*. Valid for firmware version October10, 2009.

[15] SENSIRION, The Sensor Company, March 16, 2011 [online]. http://www.sensirion.com/en/01_humidity_sensors/06_humidity_sensor_sht75.htm

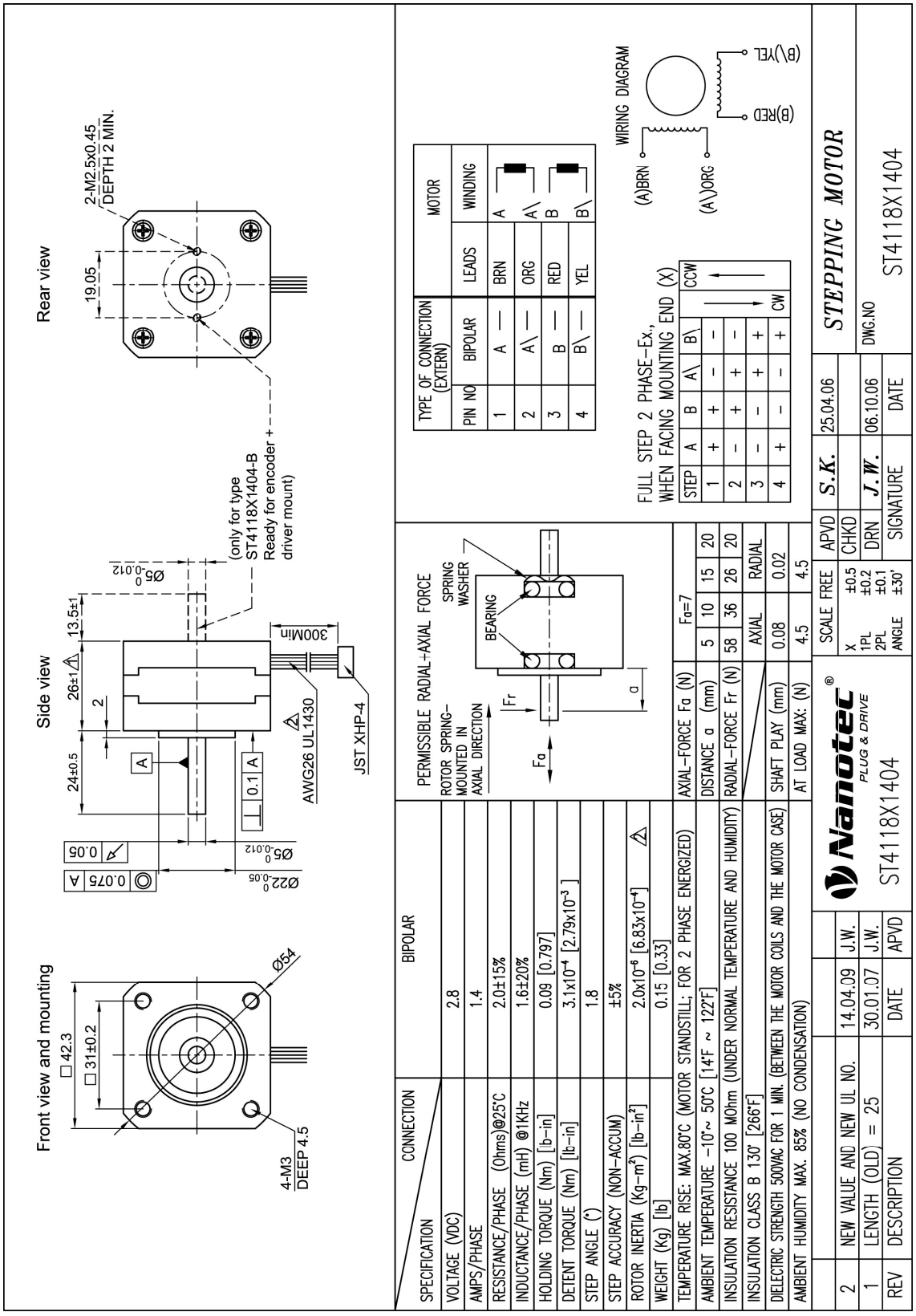
APPENDICES

Appendix 1: Datasheet of Nanotec ST4118X1404 step motor.

Appendix 2: Datasheet of Nanotec SCMI33 step motor controller.

Appendix 3: LabVIEW block diagrams of the main program.

Appendix 1: Datasheet of Nanotec ST4118X1404 step motor.



CONNECTION / SPECIFICATION	BIPOLAR
VOLTAGE (VDC)	2.8
AMPS/PHASE	1.4
RESISTANCE/PHASE (Ohms)@25°C	2.0±15%
INDUCTANCE/PHASE (mH) @1KHz	1.6±20%
HOLDING TORQUE (Nm) [lb-in]	0.09 [0.797]
DETENT TORQUE (Nm) [lb-in]	$3.1x10^{-4}$ [$2.79x10^{-3}$]
STEP ANGLE (°)	1.8
STEP ACCURACY (NON-ACCUM)	±5%
ROTOR INERTIA ($Kg-m^2$) [$lb-in^2$]	$2.0x10^{-6}$ [$6.83x10^{-4}$]
WEIGHT (Kg) [lb]	0.15 [0.33]
TEMPERATURE RISE: MAX.80°C (MOTOR STANDSTILL; FOR 2 PHASE ENERGIZED)	
AMBIENT TEMPERATURE -10~ 50°C [14°F ~ 122°F]	
INSULATION RESISTANCE 100 MOhm (UNDER NORMAL TEMPERATURE AND HUMIDITY)	
INSULATION CLASS B 130° [266°F]	
DIELECTRIC STRENGTH 500VAC FOR 1 MIN. (BETWEEN THE MOTOR COILS AND THE MOTOR CASE)	
AMBIENT HUMIDITY MAX. 85% (NO CONDENSATION)	

AXIAL-FORCE Fa (N)	Fa=7			
DISTANCE a (mm)	5	10	15	20
RADIAL-FORCE Fr (N)	58	36	26	20
	AXIAL		RADIAL	
SHAFT PLAY (mm)	0.08		0.02	
AT LOAD MAX: (N)	4.5		4.5	

X	±0.5
1PL	±0.2
2PL	±0.1
ANGLE	±30'

TYPE OF CONNECTION (EXTERN)		MOTOR	
PIN NO	BIPOLAR	LEADS	WINDING
1	A —	BRN	A
2	A\ —	ORG	A\
3	B —	RED	B
4	B\ —	YEL	B\

STEP	A	B	A\	B\	
1	+	+	-	-	CCW
2	-	+	+	-	
3	-	-	+	+	
4	+	-	-	+	CW

Appendix 2: Datasheet of Nanotec SCMI33 step motor controller.

Nanotec

Closed Loop Positioning controller with encoder input, SMCI33

Technical data

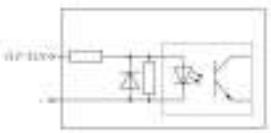
Operating voltage:	DC 24 to 48 V
Phase current:	Nominal current 2A, adjustable up to max. 3 A/phase
Interface:	RS485 or USB
Operating mode:	Position, speed, flag position, clock direction, analog, joystick
Step resolution:	1/1, 1/2, 1/4, 1/5, 1/8, 1/10, 1/32, adaptive (1/128)
Step frequency:	0 to 50kHz in the clock/direction mode, 0 to 25 kHz in all other modes
Inputs:	6 optocoupler inputs (5-24V)
Outputs:	3 transistor outputs (open collector)
Position monitoring:	Automatic error correction up to 0.9°
Current drop:	Adjustable 0- 100%
Protection circuit:	Overvoltage, undervoltage and heatsink temperature > 80 °C
Temperature range:	0 to + 40°C

* Phoenix connectors are included in the delivery.

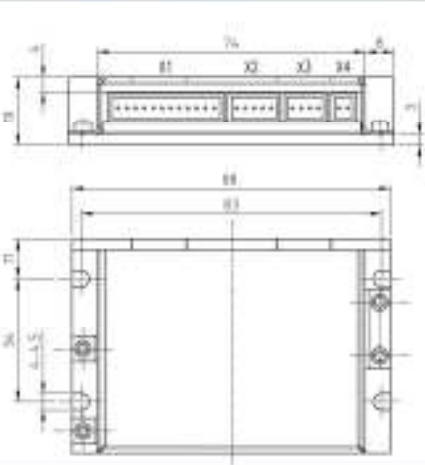
Note: A charging capacitor of at least 4,700 µF (Z-K4700/50) **must** be provided on the supply voltage so that the admissible voltage is not exceeded during the braking process.

Input circuits

Optocoupler



Outline drawing (mm)



X1-X4 Phoenix connectors
Type MICRO COMBICON
X5 Mini-USB, type B (SMCI33-1)
Sub-D 9-pole (SMCI33-2)

Inputs/Outputs (X1)

Pin	Function
1	Input1
2	Input2
3	Input3
4	Input4
5	Input5
6	Input6
7	Com
8	Output 1
9	Output 2
10	Output 3
11	Analogue in
12	GND

Encoder (X2)

Pin	Function
1	+5V
2	CH B
3	CH A
4	INDEX
5	GND

Motor connection (X3)

Pin	Function
1	Motor coil A
2	Motor coil A\
3	Motor coil B\
4	Motor coil B

Supply (X4)

Pin	Function
1	UB24-48V
2	GND

SMCI33-2: RS485 (X5)

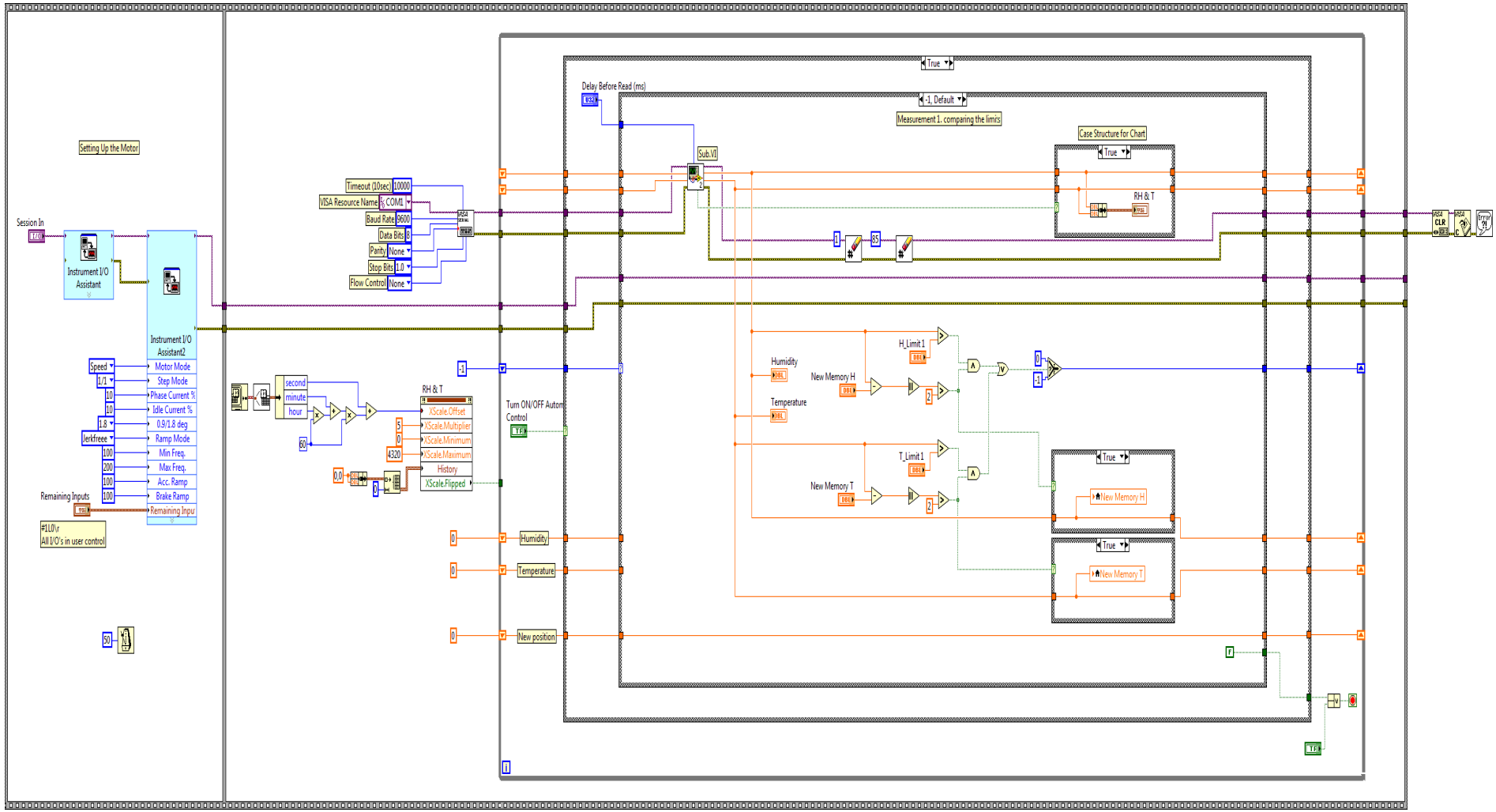
Pin	Function
1	NC
2	RX+
3	+5V
4	TX+
5	N.C.
6	N.C.
7	RX-
8	GND
9	TX-

SMCI33-1: USB (X5)
USB standard

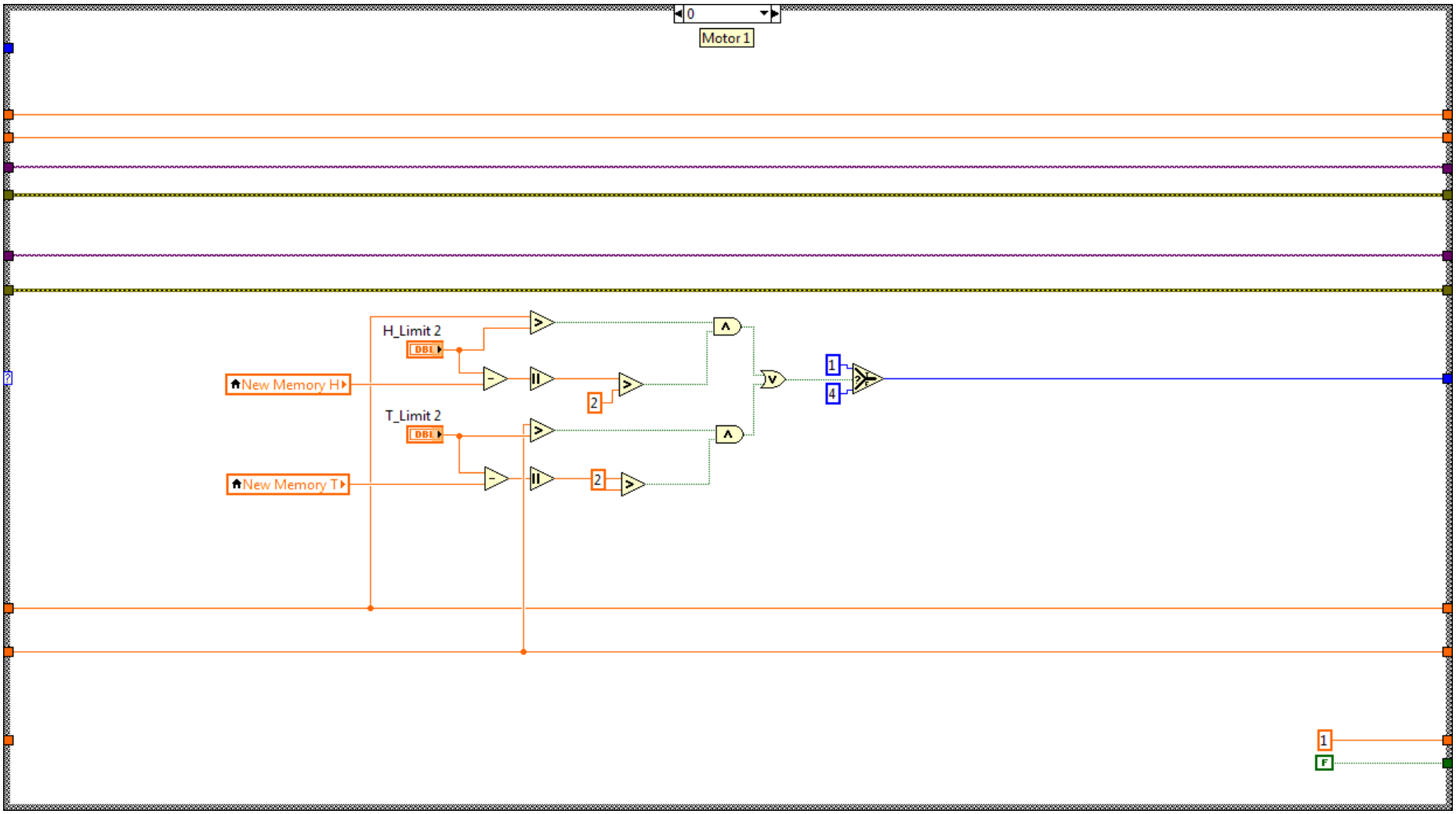
Order number

SMCI33 -
1 = USB
2 = RS485

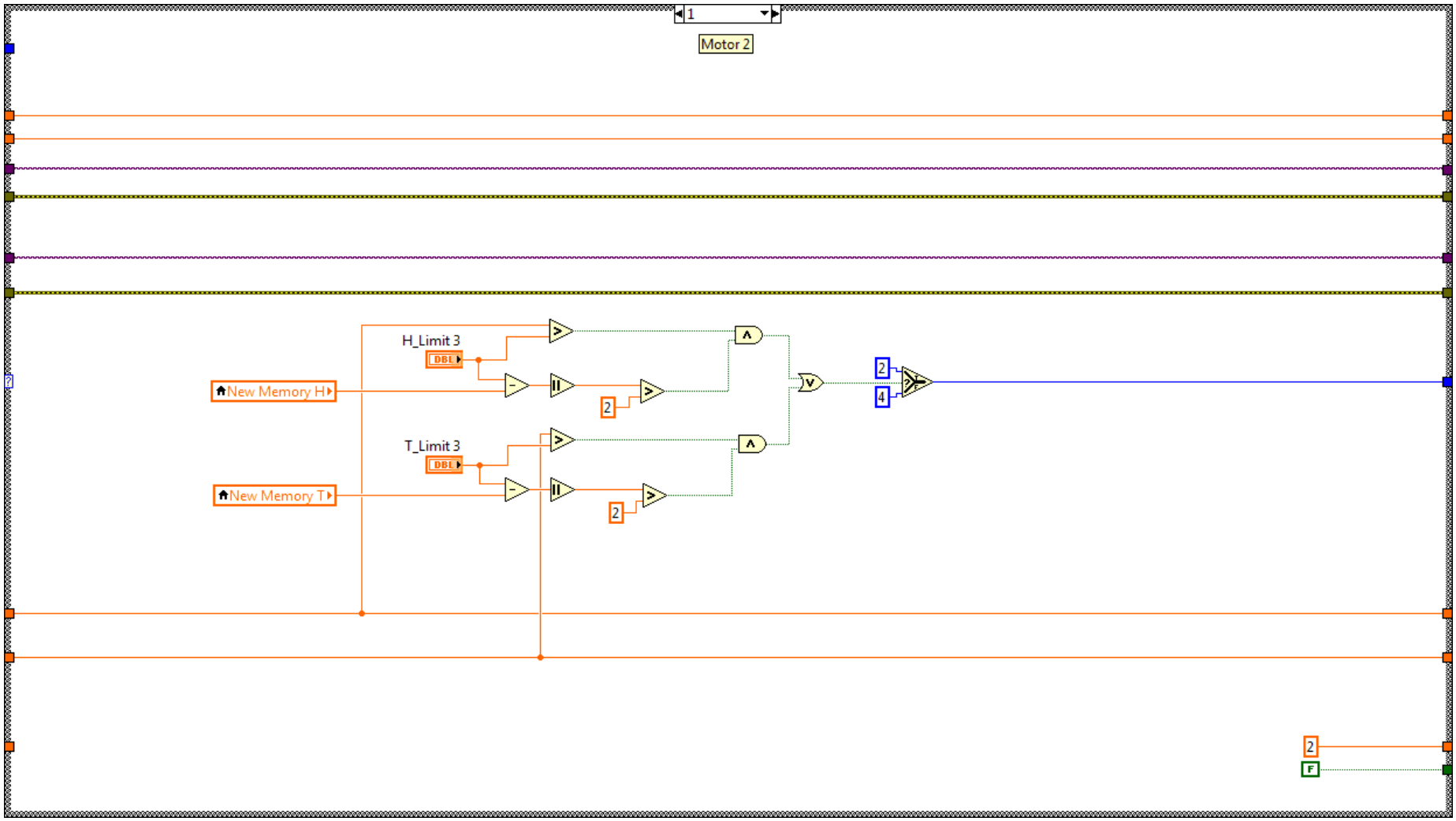
Appendix 3 (1): This is the first page of the main program and the true case structure of the turn on/off automatic control. This is also the first limit comparison subdiagram of the program. The functionality of this subdiagram is to either pass the program to the next subdiagram or to keep the program here to go through this subdiagram's operations again based on the outcomes.



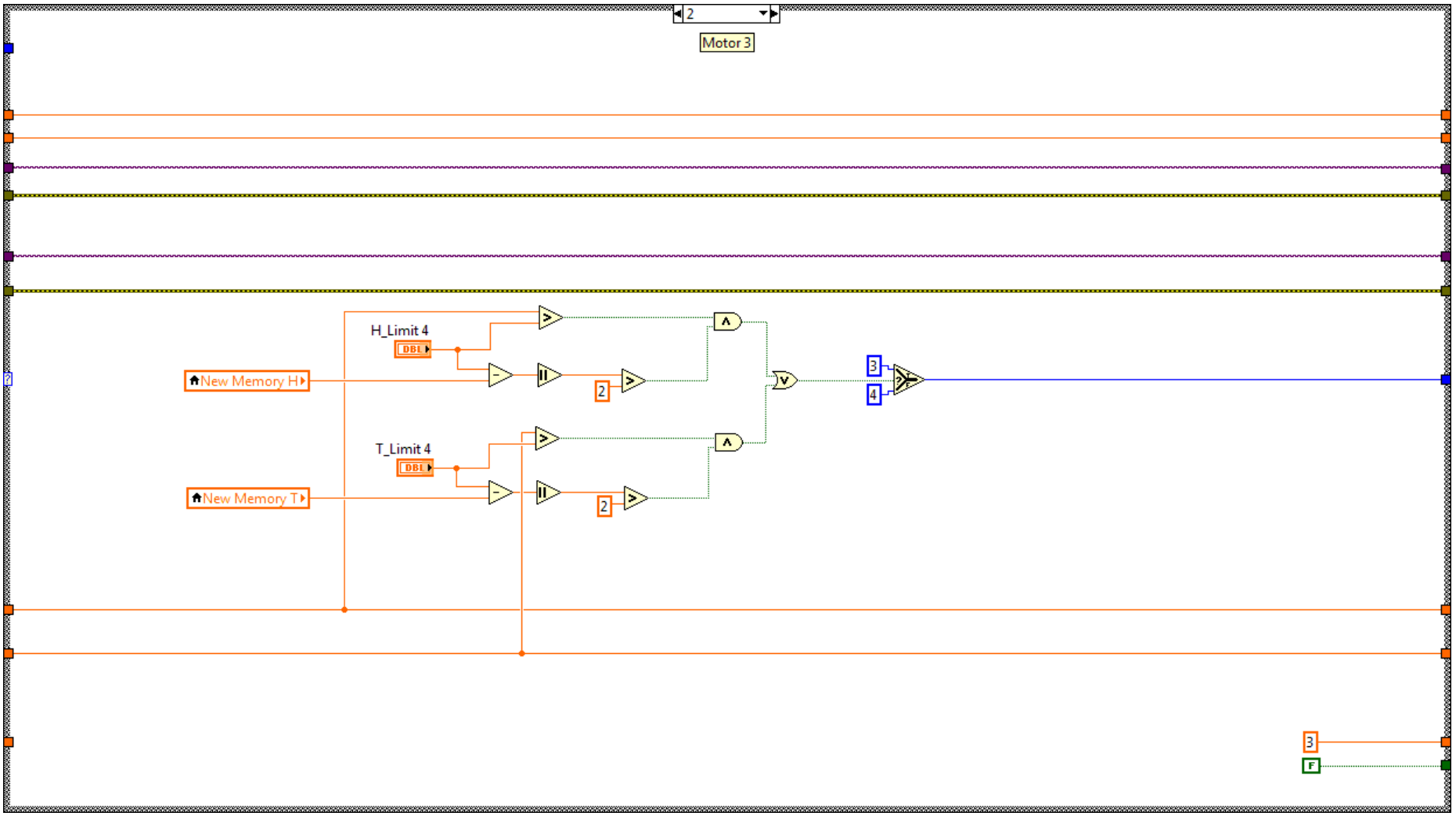
Appendix 3 (2): This is the second limit comparison subdiagram. The functionality of this subdiagram is to either pass the program to the next subdiagram or to the motor control subdiagram to set the motor at position 1 based on the outcomes.



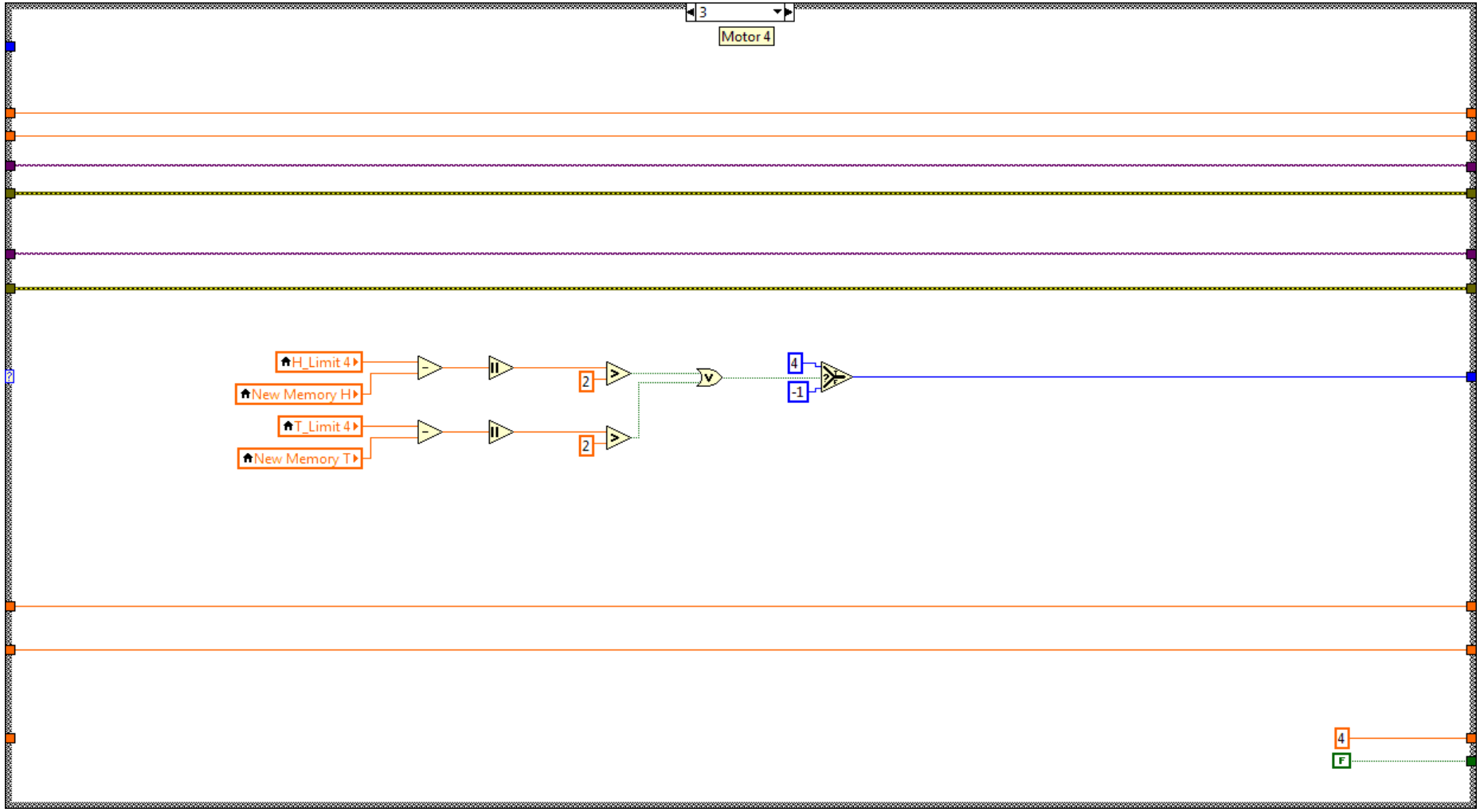
Appendix 3 (3): This is the third limit comparison subdiagram. The functionality of this subdiagram is to either pass the program to the next subdiagram or to the motor control subdiagram to set the motor at position 2 based on the outcomes.



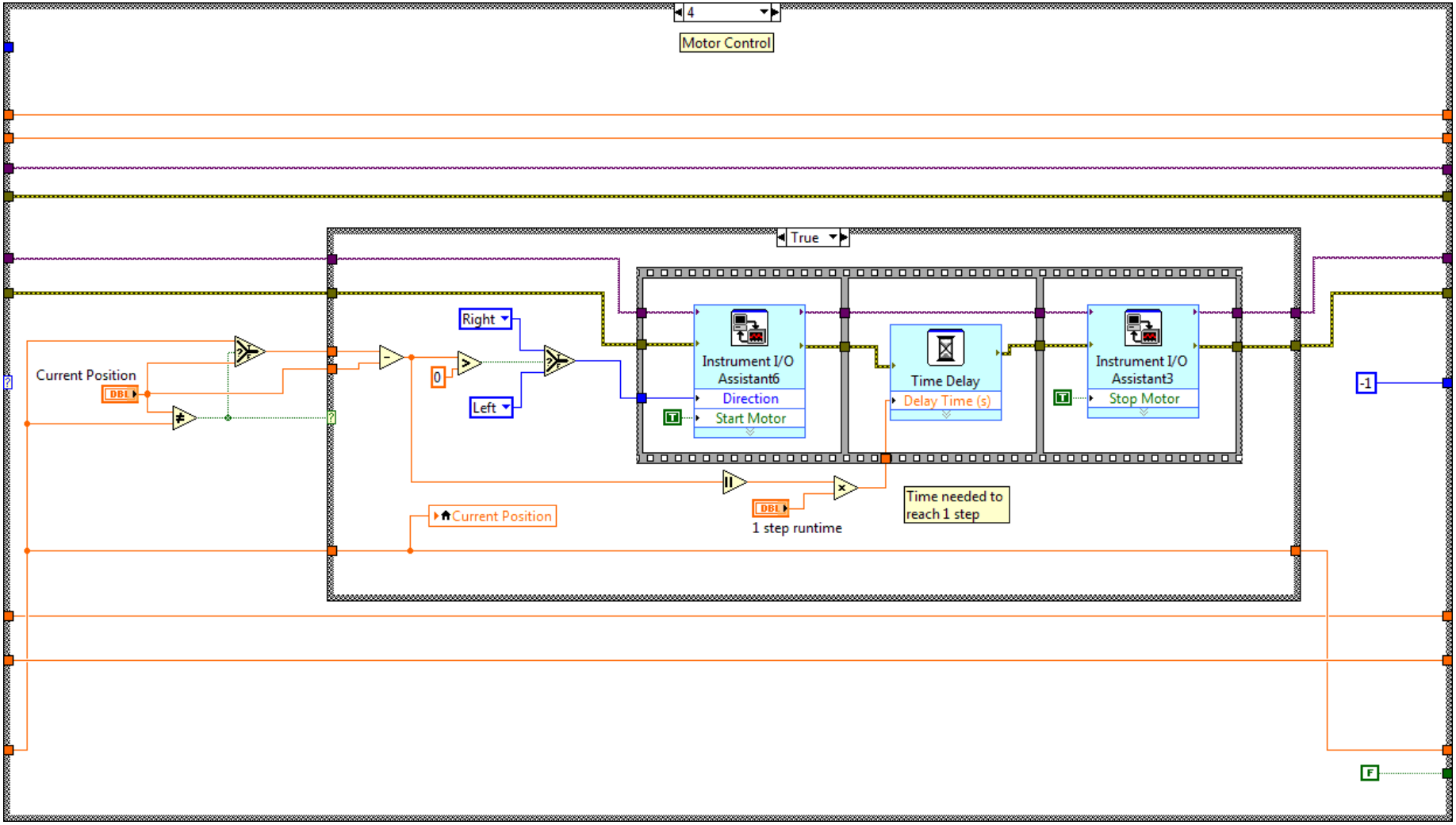
Appendix 3 (4): This is the fourth limit comparison subdiagram. The functionality of this subdiagram is to either pass the program to the next subdiagram or to the motor control subdiagram to set the motor at position 3 based on the outcomes.



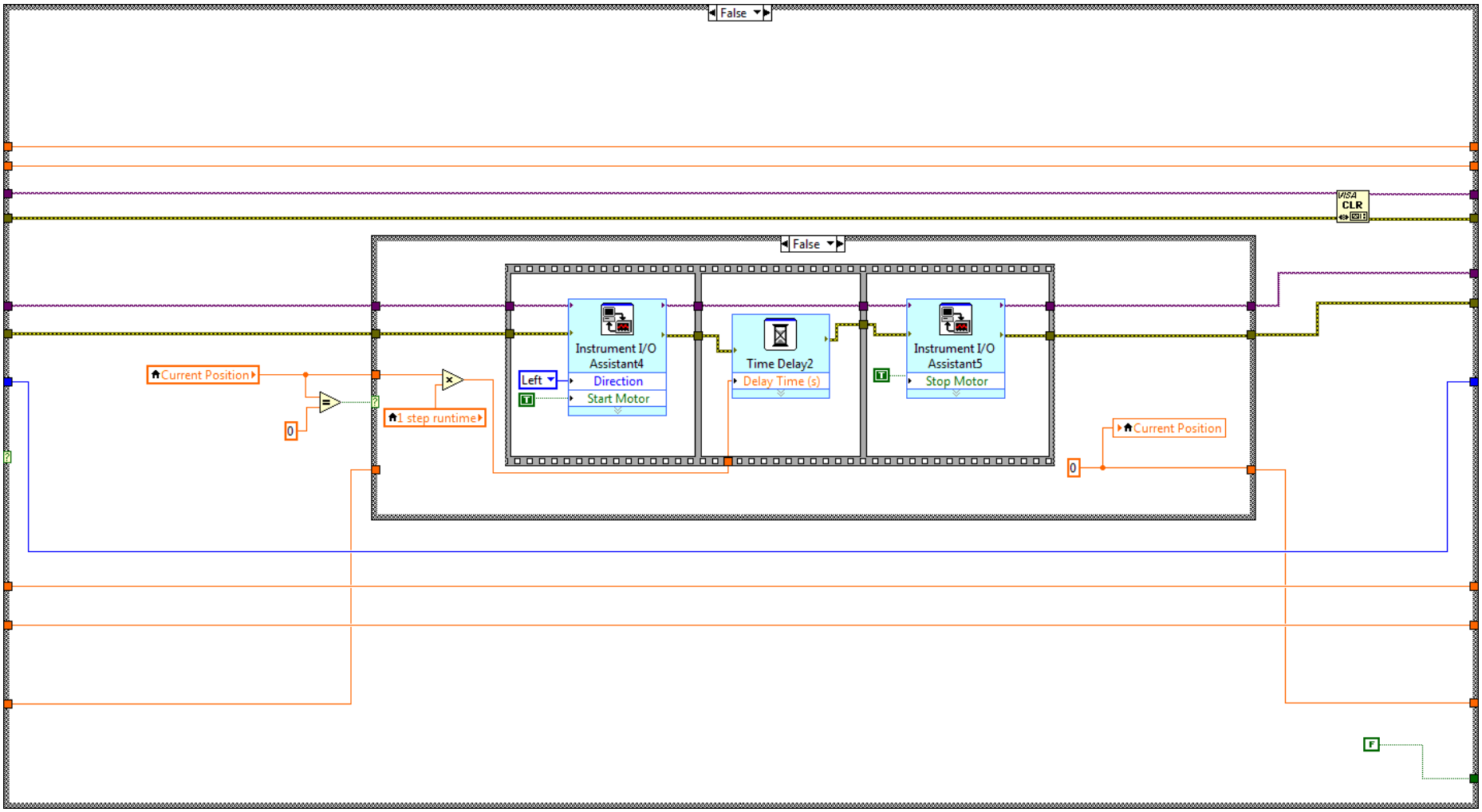
Appendix 3 (5): This is the subdiagram to check if there has been enough change between the fourth limits and the new readings. The functionality of the program is to either pass the program to the motor control subdiagram to set the motor at position 4 or to the first comparison surbdiagram based on the outcomes.

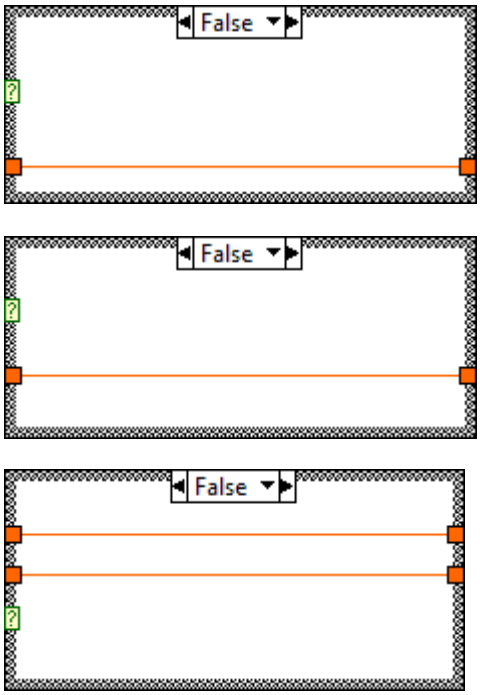


Appendix 3 (6): This is the motor control subdiagram. The functionality of this subdiagram is to turn on the motor and set the motor at the right position based on the outcomes and turn off the motor after adjusting.



Appendix 3 (7): This is the false case structure of the turn on/off automatic control and the false case structure of the set to zero case structure. The functionality of this sudbiagram is to set the motor at position 0 and turn off the automatic control if the turn on/off automatic control button is turned off.





Appendix 3 (8): These are the false case structures of the humidity and temperature new memory and the history chart. Here are also the true case structure of the turn on/off automatic control and the false case structure of the motor control. The functionalities of these structures are to just pass the data through.