DDU SMART HOME

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Web based smart home is a concept that makes home automation online. It allows user to control and monitor their homes via web sitting anywhere in the world. We have included many features in our home model and a lot more can be added. Some of this features are automated and manual operation of lights and fan, smoke detection, music system in washroom, speed control of fan depending on temperature, fire alert and mail alert by Email, main door security, intruder alert through Email with attachment of intruder's photo, etc.

All this is done by Arduino and LabVIEW. Arduino as controller and LabVIEW as user interface. Also unlike SCADA, most of the programming is done in LabVIEW. Graphical programming of LabVIEW makes it simpler and easier to understand. User can also change some features by himself if he has some knowledge of LabVIEW. LIFA (LabVIEW interface for Arduino) is used to interface LabVIEW and Arduino. LIFA is perhaps the most important concept that helped us in making our project successful. Web publishing tool of LabVIEW is used to publish the VI online which any user can control and operate, provided that he/she has correct password. Many clients can join to server. Server computer is the one in which all VI's are made and is connected to the system. At a time, either server or client can control the system. However server can kick client anytime. Remote desktop app of google can be used to connect to the server directly from any device connected to net from anywhere in the world. By this we can say that the operator acts as remote server.

Web based home automation is still in its maturing stage and a lot can be expected in near future.

1 INTRODUCTION

1.1 INTRODUCTION TO WEB BASED SMART HOME

Web based smart home as the name suggest is home automation which can be controlled via web. We have tried to automate many features of house. We have controlled lights, fan, fire or smoke detection, front door security, CCTV footage, letter box and music control.

Most of the features can be controlled both automatically and manually. Automatic and manual operation can be easily changed by owner of the house and those who have access to VI which is password protected too. The system in which original VI is made is called server and it has to remain inside house connected to LAN. Server has full control of VI gaining all access and clients which are members of the house can join in to server and request control of the VI from server.

Apart from this using remote desktop app anyone can join in to server and control server computer. This is password protected too. So we can control and monitor our home from every corner of the world provided that we have good net access.

The user interface is simple to understand and easy to operate. **Tab control** in front panel provides better way of maneuvering through different parts of house. We have a hall, kitchen, bedroom, washroom and letter box. Apart from letter box all parts has light which operates in presence of person detected by PIR. Hall has a fan too whose speed is controlled by temperature. LM35 senses the temperature. Besides this automatic operation both light and fan have been included in manual operation too. To switch on light or fans all u need to do is switch it on your mobile and let the magic happen. This can prove to be a virtue for old people who find difficulty in day to day life. Kitchen has smoke detection system apart from light. MQ135 air quality sensor is used for smoke detection and whenever it detects smoke, buzzer will turn on thus alarming people inside house. Also the owner of the house will receive a fire alert via mail. So in case nobody is at home still the owner of the house will receive mail and so he/she can take appropriate steps.

Many people like to hear music while taking shower. So we have included this entertainment feature in our house too. As soon as you turn on the shower music will be played. Flow switch should be used ideally to detect shower but due to hardware constraint we have used a toggle switch to indicate shower.

Main door is secured by a password. Keypad is provided to write a password. If the user enters correct password door will open. Door will remain open till 20 sec and then it will close again making it mandatory to write correct password again to open it. If the user enters wrong

password for three times his/her picture will be taken and sent to the owner of the house in attachment via mail. Mail subject will be intruder alert which can obviously be changed from VI.

Letter box is having a limit switch placed in such a way that whenever a letter in dropped into letter box it trips. So when a letter is dropped, mail alert will be sent to the owner of the house. All mails will include date and time of the event happening so that the owner can know when it has happened.

Web publishing, remote desktop and mail alerts are some of crucial features of our project. Also this has entirely been done by LabView and Arduino. Both of this are easily available without much cost. Costlier parts are relay module, motor driver and certain sensors. but overall entire cost of the idea is low and easily affordable.

Many features can still be added in this and also this can improved. Due to financial and other constraints all features cannot be included in this. However following are certain features that can be easily installed with our system:

- Drape control in bed room
- Heater/AC control
- Better burglar system
- Energy meter
- Any application of solar energy

Not that we haven't tried but inclusion of every feature is hard when seen from financial point of view.

All in all making this web based was our main objective and so we are excited about it.

2 LITERATURE STUDY

2.1 LABVIEW

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) is a Platform and development environment for a visual programming language from National Instruments. LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. LabVIEW contains a comprehensive set of tools for acquiring, analyzing, displaying, and storing data.

LabVIEW VI's contain three components—the front panel, the block diagram, and the icon and connector pane.

In LabVIEW, you build a user interface, or front panel, with controls and indicators. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. After you build the user interface, you add code using VIs and structures to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

Use LabVIEW to communicate with hardware such as data acquisition, vision, and motion control devices, and GPIB, PXI, VXI, RS-232, and RS-485 devices. LabVIEW also has built-in features for connecting your application to the Web using the LabVIEW Web Server and software standards such as TCP/IP networking and ActiveX.

LabVIEW is a system-design platform and development environment for a visual programming language from National Instruments.

There are two types of programming:

Dataflow programming

Graphical programming

Dataflow programming

Execution of this is determined by the structure of a graphical block diagram (the LabVIEW-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution.

Graphical programming

LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel and a connector panel. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators.

The graphical approach also allows non-programmers to build programs by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and documentation, makes it simple to create small applications.

Benefits

- LabVIEW includes extensive support for interfacing to devices, instruments, cameras, and other devices.
- In terms of performance, LabVIEW includes a compiler that produces native code for the CPU platform.
- The graphical code is translated into executable machine code by interpreting the syntax and by compilation.
- Many libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous graphical interface elements are provided in several LabVIEW package options.

Applications

- Acquiring Data and Processing Signals
- Instrument Control
- Automating Test and Validation Systems
- Embedded Monitoring and Control Systems
- Academic Teaching
- Wireless System Prototyping
- Sophisticated Biomedical Tissue Measurement Using Image Analysis and Virtual Instrumentation
- Acoustical Measurement and Fan Fault Diagnosis System Based on LabVIEW
- Digital Image Processing Using LabView
- Application of Virtual Instrumentation in Nuclear Physics Experiments

Examples of different NI LabVIEW Toolkits

- Advanced analysis functions (wavelets, joint time-frequency analysis)
- Application-specific analysis (sound and vibration, order analysis, spectral)
- New and industry-standard technologies (DSP, OPC, PID, simulation)
- Off-the-shelf data management for test files
- Data logging, alarms, security, and machine vision
- Report generation and connectivity (SQL, ODBC, Microsoft Excel)

2.1.1 Creating VI

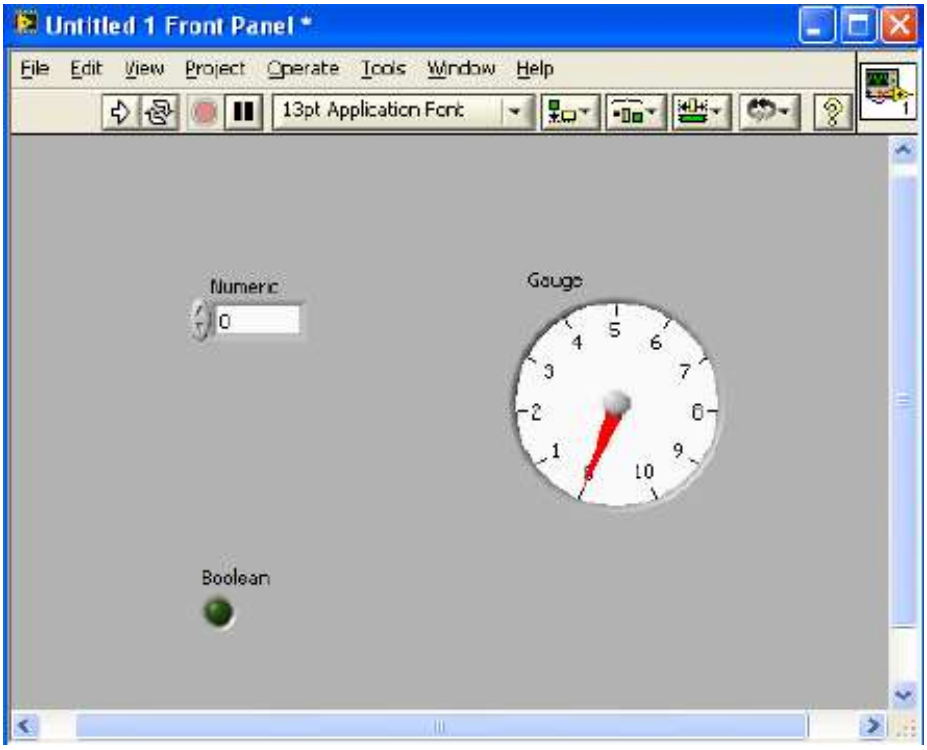
When you open a blank VI, an untitled front panel window appears. This window displays the front panel and is one of the two LabVIEW windows you use to build a VI. The other window contains the block diagram. Every front panel has at most one block diagram, and every block diagram has exactly one front panel. The front panel is what the user sees (sometimes called a GUI, or graphical user interface), and the block diagram is the code, or the heart of the program.

Front panel

In LabView, you build the front panel with controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

Block diagram

After you build the user interface, you add code using VI's and structures to control the front panel objects. The Block diagram contains this code. In some ways, the block diagram resembles a flowchart. After you build the front panel, you add code using graphical representations of functions to control the front panel objects. The Block diagram contains this graphical source code. Front Panel objects appear as terminals, on the block diagram. Block Diagram objects include terminals, sub VI's, functions, constants, structures, and wires, which transfer data among other block diagram objects.



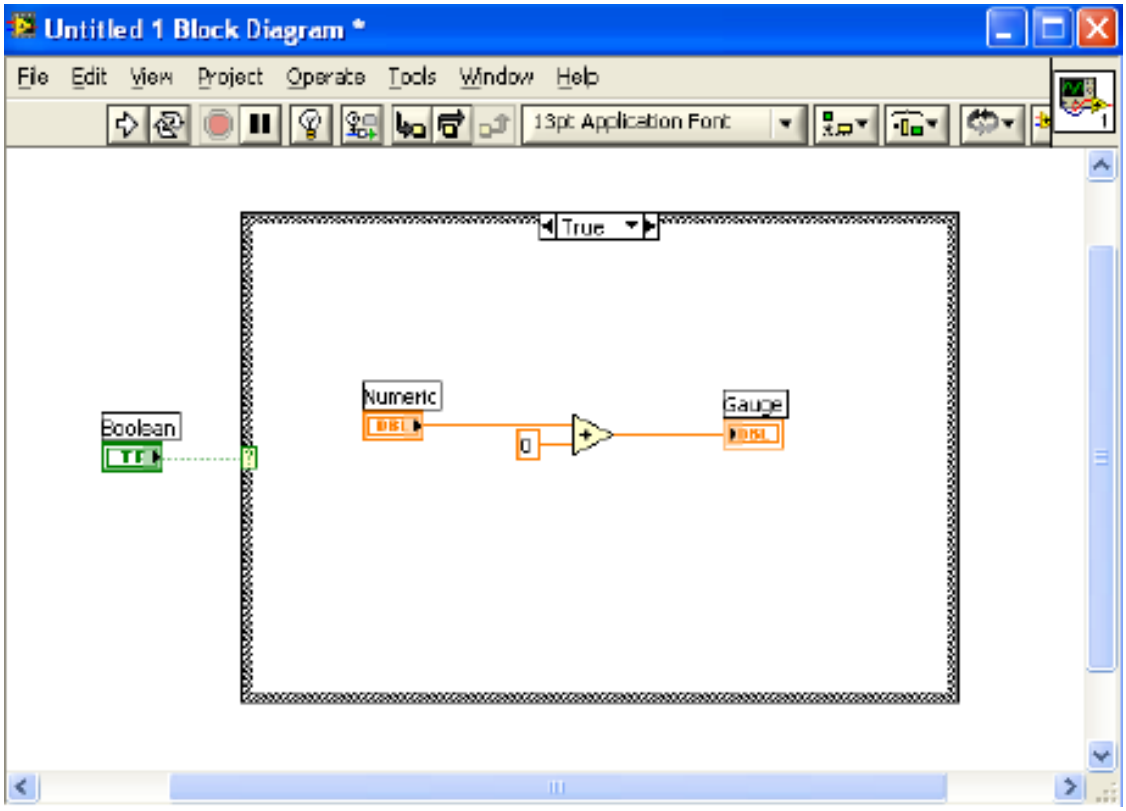


FIG 2.1 Example of Front panel and block diagram

Example:
Below is example of a simple VI which is used to get current DATE AND TIME. This VI has been used in project as a sub VI. The front panel and block diagram of VI is shown below:

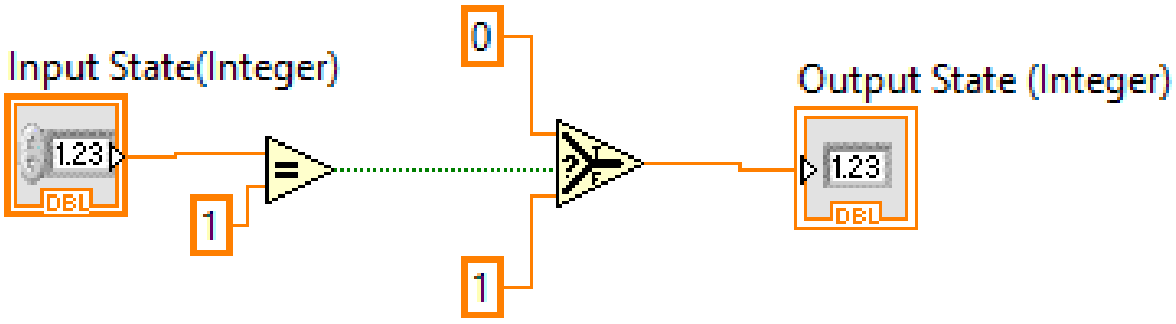


FIG 2.2 Example of block diagram

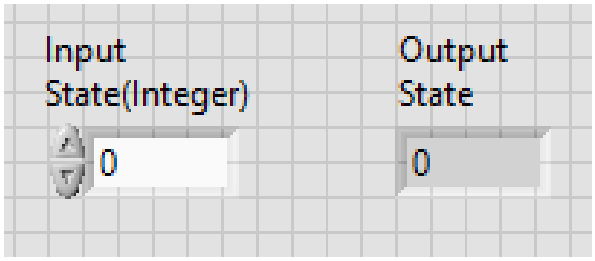


FIG 2.3 Example of front panel

Fig shows block diagram and front panel. Controls palette is used to make VI.

2.1.2 Controls palette

The Controls and Functions Palettes contain sub palettes of objects you can use to create a VI. When you click a sub palette icon, the entire palette changes to the sub palette you selected. To use an object on the palettes, click the object and place it on the front panel or block diagram. The Controls Palette is available only on the front panel. The Controls Palette contains the controls and indicators you use to build the front panel.

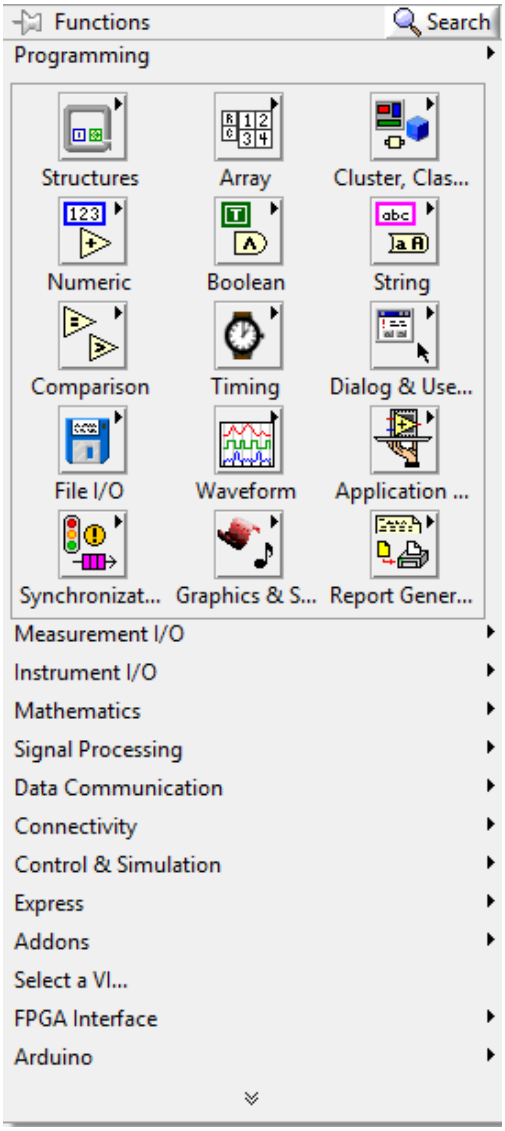


FIG 2.4 Control palette

The most used Sub Palettes are the Numeric Sub Palette, The Boolean Sub Palette and the String & Path Sub Palette.

Following are few most commonly used palettes:
Palettes shown below are from LabView version we have used.

Numeric sub palette

"Numerical Control" And "Numerical Indicator" are the most used objects in the numeric sub palette.

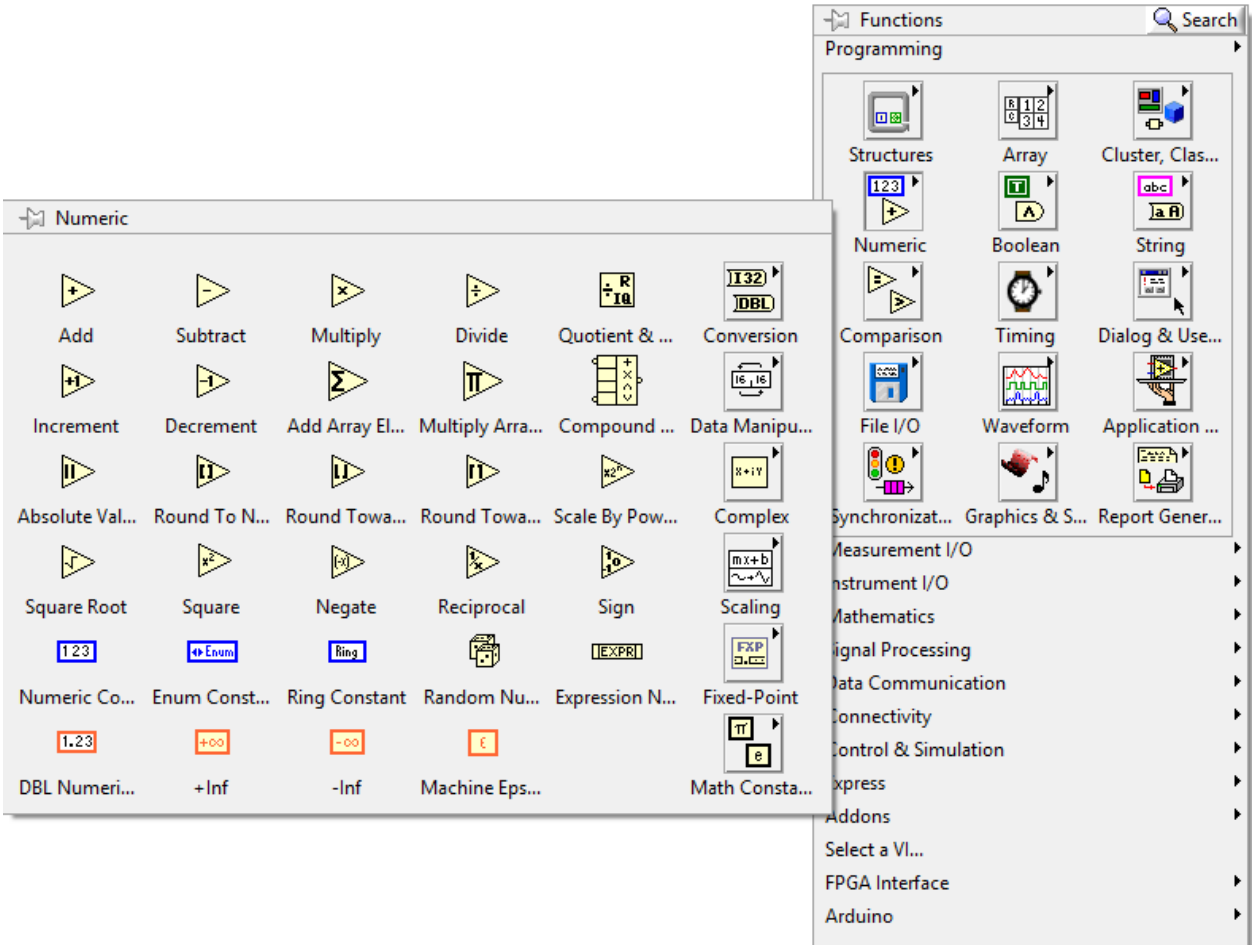


FIG 2.5 Numeric sub palette

Boolean sub palette

This palette has lots of different buttons you may use. OK, Cancel and Stop Buttons are useful.

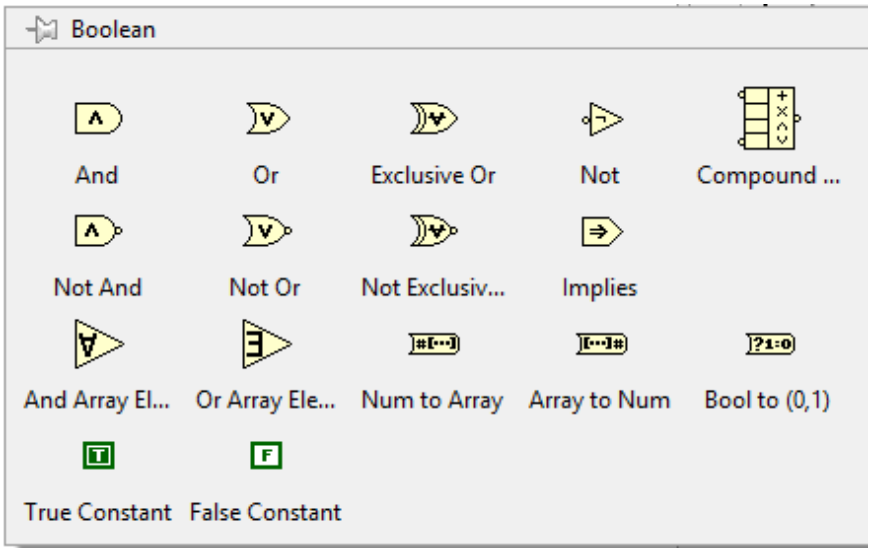


FIG 2.6 Boolean sub palette

String and path sub palette

Concatenate strings is the object we have used most from strings and path sub palette.

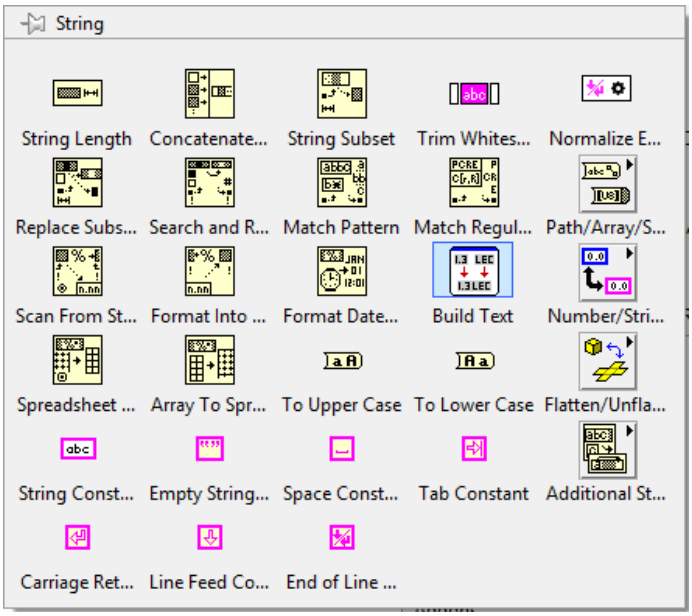


FIG 2.7 String and path sub palette

2.1.3 Sub VI'S

The power of LabVIEW lies in the hierarchical nature of the VI. After you create a VI, you can use it on the block diagram of another VI. There is no limit on the number of layers in the hierarchy. Using modular programming helps you manage changes and debug the block diagram quickly.

A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages. When you double-click a subVI, a front panel and block diagram appear, rather than a dialog box in which you can configure options. The front panel includes controls and indicators that might look familiar. The block diagram includes wires, front panel icons, functions, possibly subVIs, and other LabVIEW objects that also might look familiar.

The upper right corner of the front panel and block diagram displays the
icon for the VI. This icon is the same as the icon that appears when you place
the VI on the block diagram.

We have used **9 sub VI's** in our entire LabVIew programming of web based home automation. These sub VI are as follows:

- Date Time
- Digital Read
- Digital Write
- Email
- Integer NOT gate
- Night Day
- Password Condition
- PIR
- Path

2.1.4 Loops and structures

Structures are graphical representations of the loops and case statements of text-based programming languages. Use structures on the block diagram to repeat blocks of code and to execute code conditionally or in a specific order. LabVIEW includes the following structures: structures:

- For loop
- While loop
- Case structure
- Stacked sequence structure
- Flat sequence structure
- Event structure
- Formula node

Of this loops and structures we have used for loop, while loop, case structure and flat sequence structure.

The different Loops and Structures available are located in the "*Structures*" Sub palette in the Functions Palette on the Block Diagram.

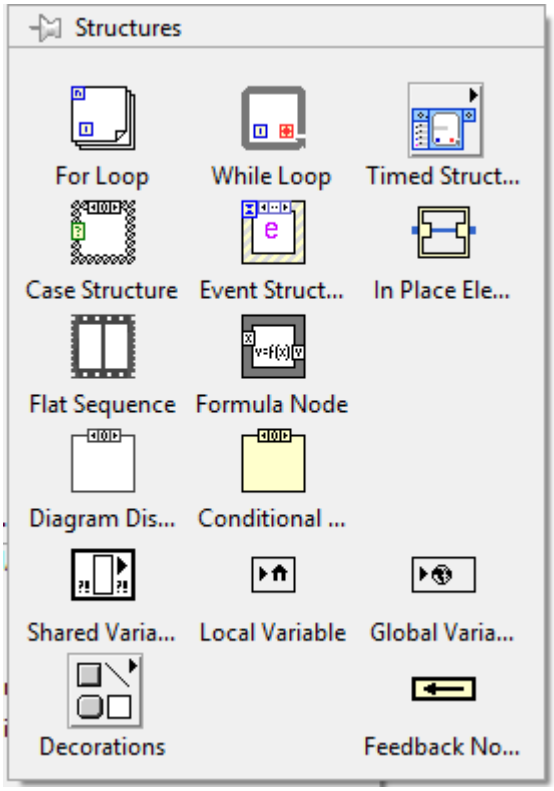


FIG 2.8 Structures

For loop

A For Loop executes a sub diagram a set number of times. The figure below shows an empty For Loop In LabVIEW.

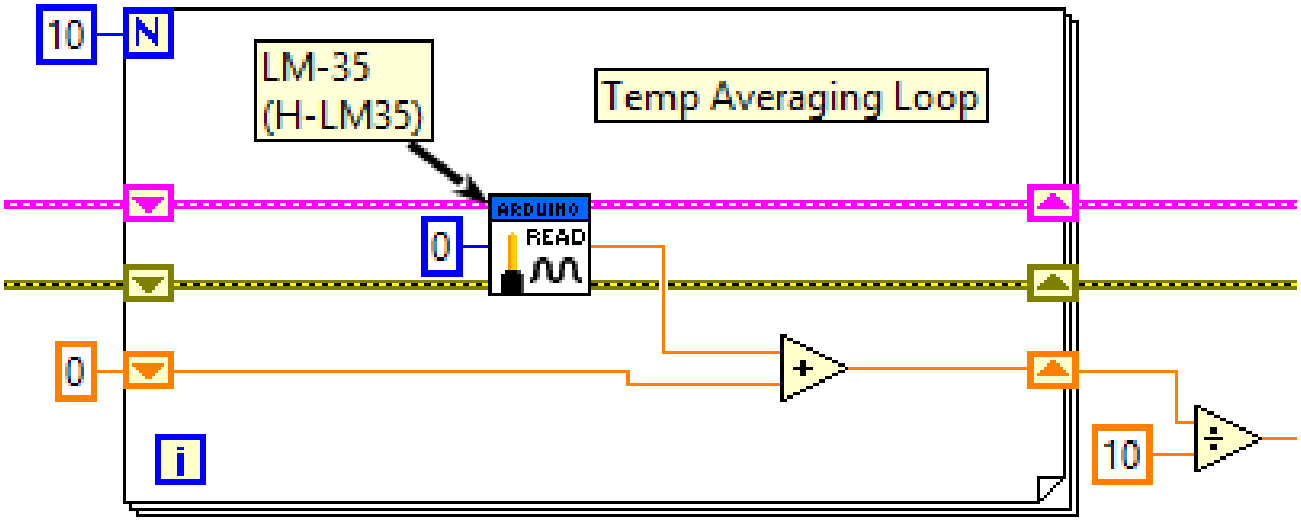


FIG 2.9 Empty for loop

Case structure

The Case Structure has one or more sub diagrams, or cases, exactly one of which executes when the structure executes. The value wired to the selector terminal determines which case to execute and can be boolean, string, integer, or enumerated type. You may right---click the structure border to add or delete cases. Use the Labeling Tool to enter value(s) in the case selector label and configure the value(s) handled by each case.

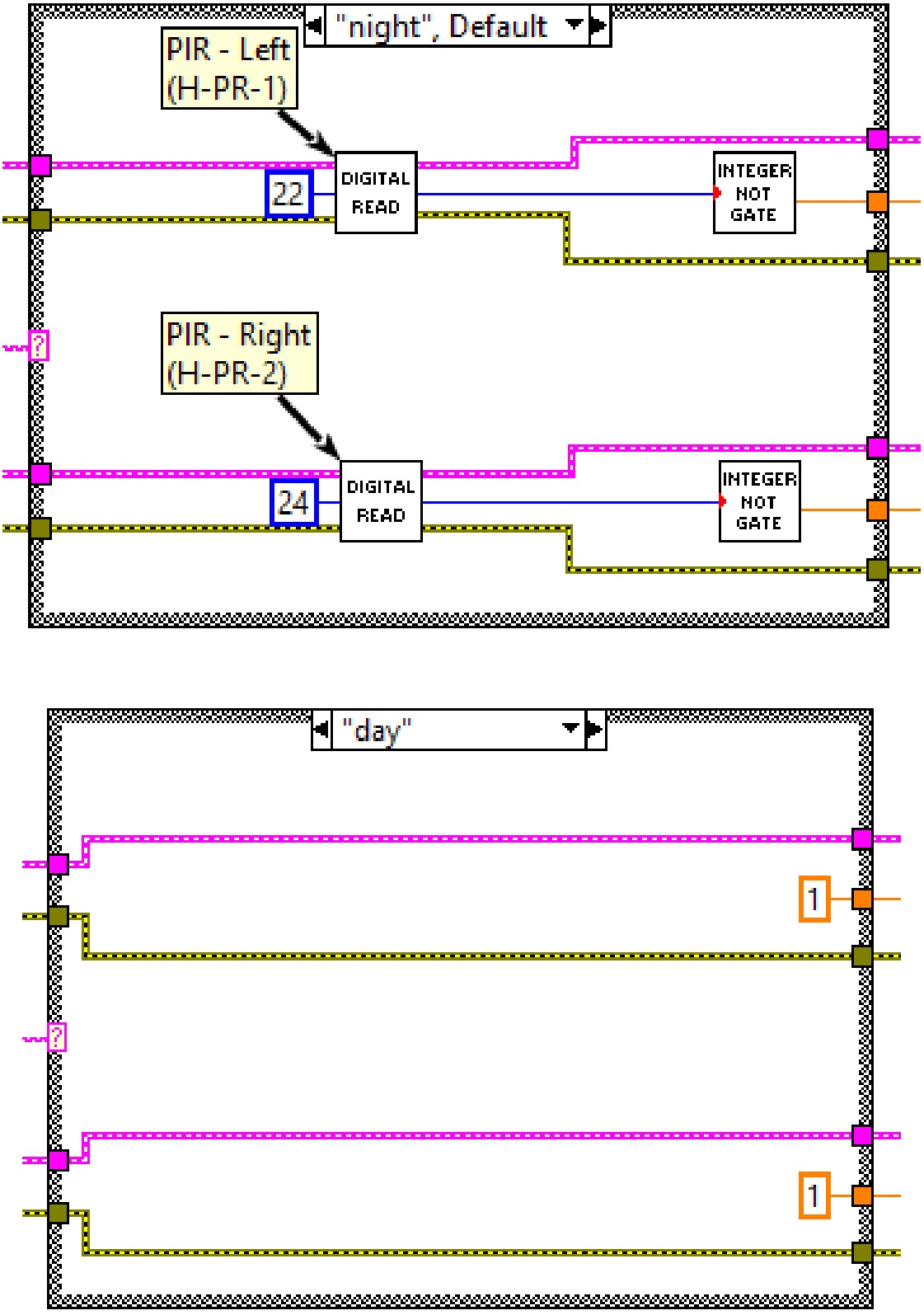


FIG 2.10 Case structure

Flat sequence structure

Consists of one or more subdiagrams, or frames, that execute sequentially. Use the Flat Sequence structure to ensure that a subdiagram executes before or after another subdiagram.

Data flow for the Flat Sequence structure differs from data flow for other structures. Frames in a Flat Sequence structure execute from left to right and when all data values wired to a frame are available. The data leaves each frame as the frame finishes executing. This means the input of one frame can depend on the output of another frame.

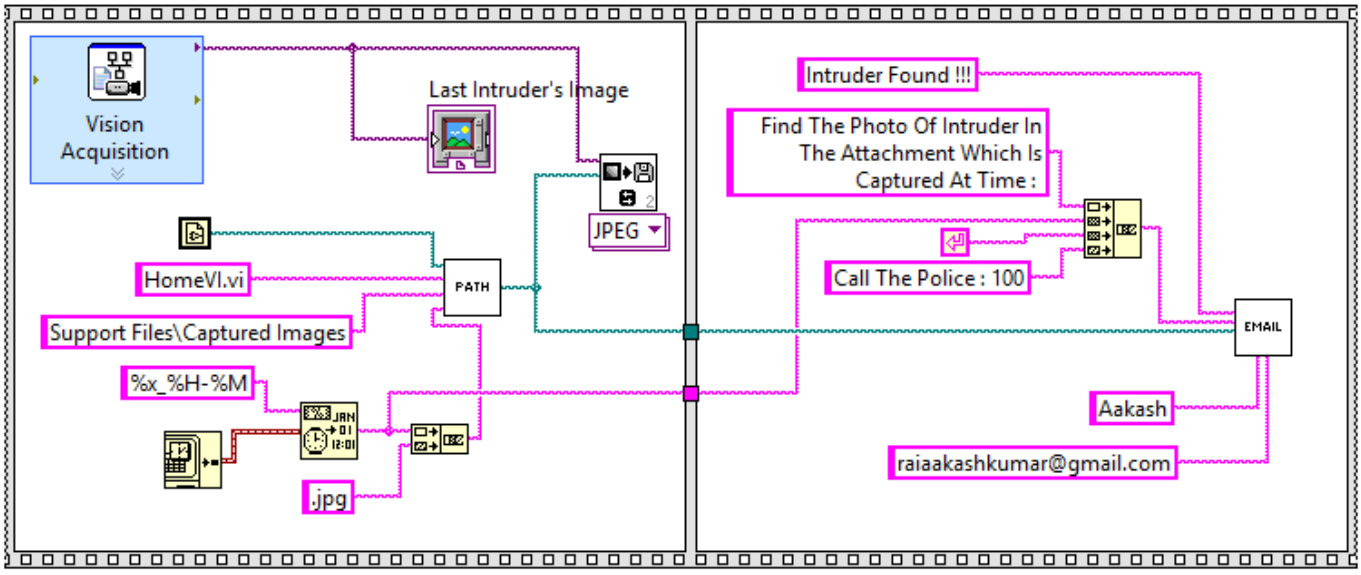


FIG 2.11 Flat sequence structure (See Appendix Fig-A)

2.1.5 Working with data

Array

Arrays group data elements of the same type. An array consists of elements and dimensions. Elements are the data that make up the array. A dimension is the length, height, or depth of an array. An array can have one or more dimensions and as many as $2^{31} - 1$ elements per dimension, memory permitting.

You can build arrays of numeric, Boolean, path, string, waveform, and cluster data types. Consider using arrays when you work with a collection of similar data and when you perform repetitive computations. Arrays are ideal for storing data you collect from waveforms or data generated in loops, where each iteration of a loop produces one element of the array.

You cannot create an array of arrays. However, you can create an array of clusters where each cluster contains one or more arrays. Array elements are ordered. An array uses an index so you can readily access any particular element. The index is zero-based, which means it is in the range 0 to $n - 1$, where n is the number of elements in the array.

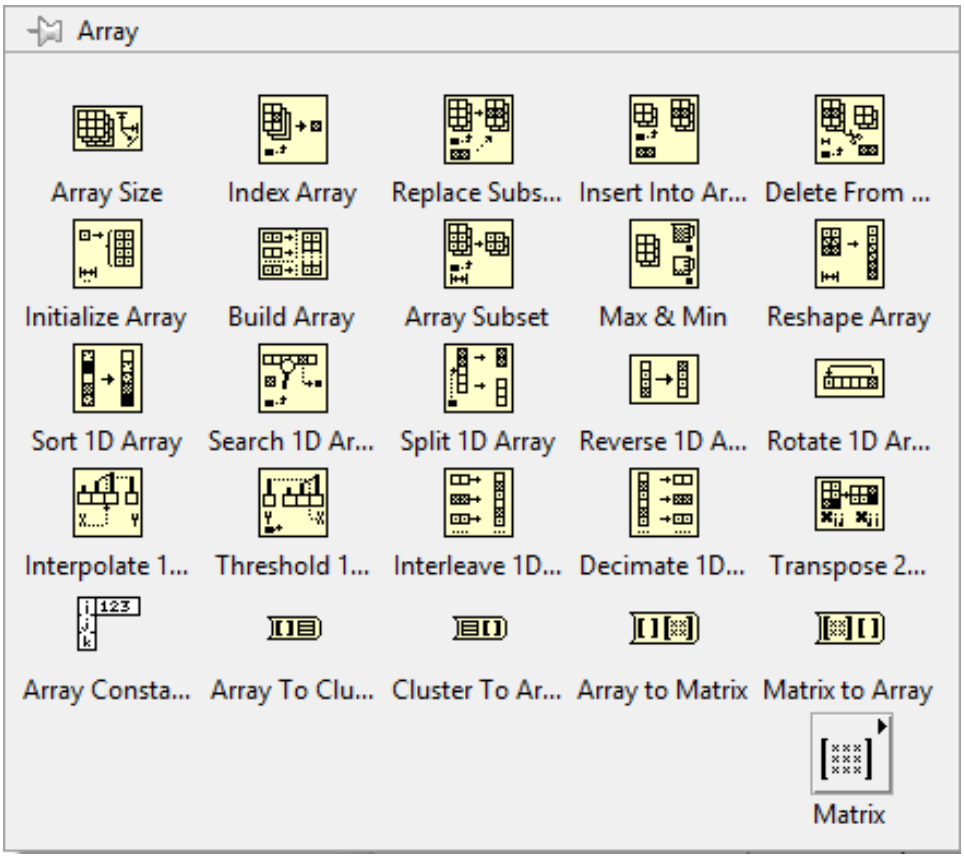


FIG 2.12 Array palette

Clusters

Clusters group data elements of mixed types, such as a bundle of wires, as in a telephone cable, where each wire in the cable represents a different element of the cluster. A cluster is similar to a record or a struct in text-based programming languages.

Bundling several data elements into clusters eliminates wire clutter on the block diagram and reduces the number of connector pane terminals that sub VIs need. Like an array, a cluster is either a control or an indicator. A cluster cannot contain a mixture of controls and indicators.

Although cluster and array elements are both ordered, you must unbundle all cluster elements at once rather than index one element at a time. You also can use the unbundle by name function to access specific cluster elements.

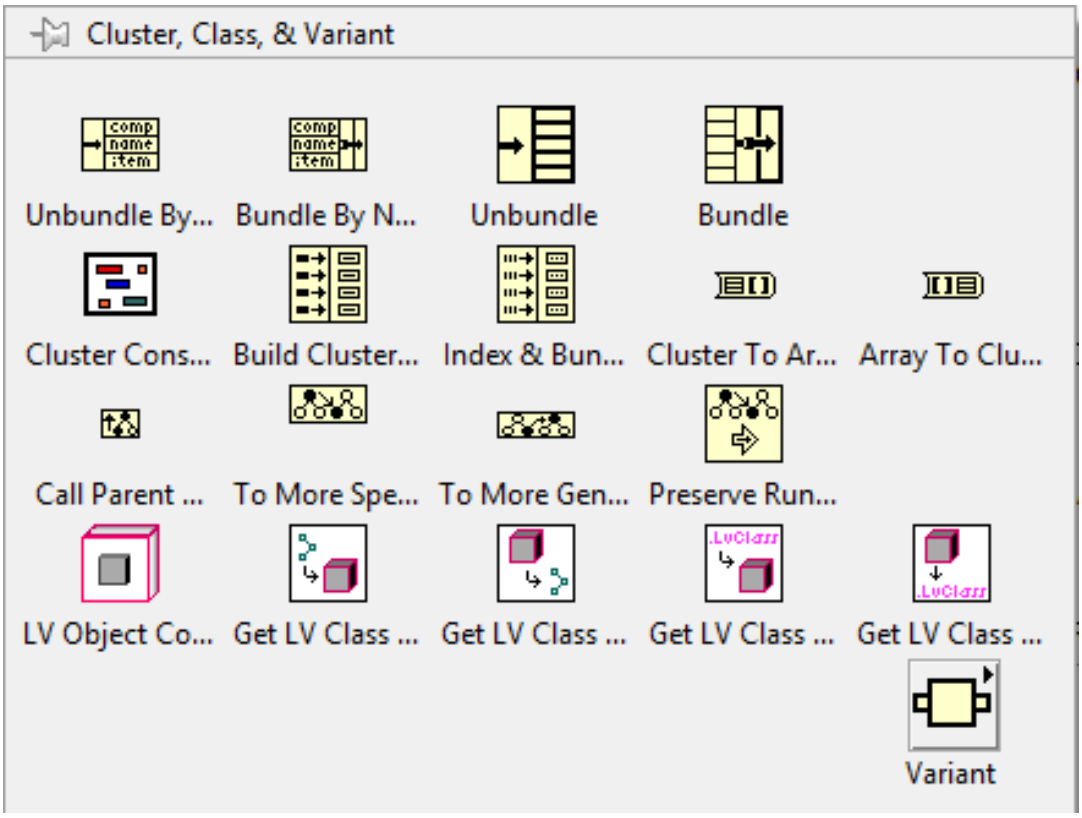


FIG 2.13 Clusters palette

Strings

A string is a sequence of displayable or non-displayable ASCII characters. Strings provide a platform-independent format for information and data. Some of the more common applications of strings include the following:

- Creating simple text messages.
- Passing numeric data as character strings to instruments and then converting the strings to numeric values.
- Storing numeric data to disk. To store numeric values in an ASCII file, you must first convert numeric values to strings before writing the numeric values to a disk file.
- Instructing or prompting the user with dialog boxes.

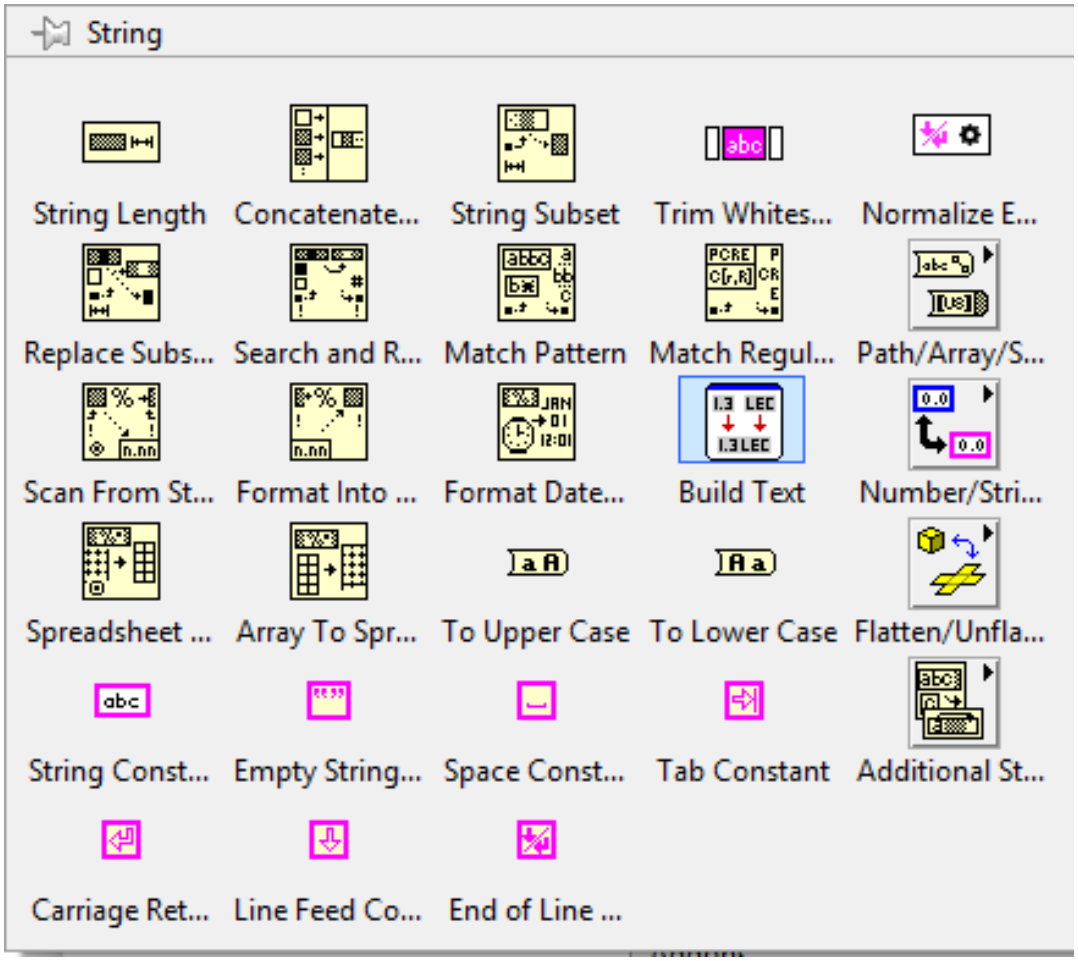


FIG 2.14 String palette

2.1.6 User interface

All the best programming is of no use if its not presented well. So making a good user interface which can be easily understood should be a first priority. LabView offers many features that are used to make better user interface. Following are some of it:

- Decorations
- Tab control
- Splitter
- Sub panel etc.

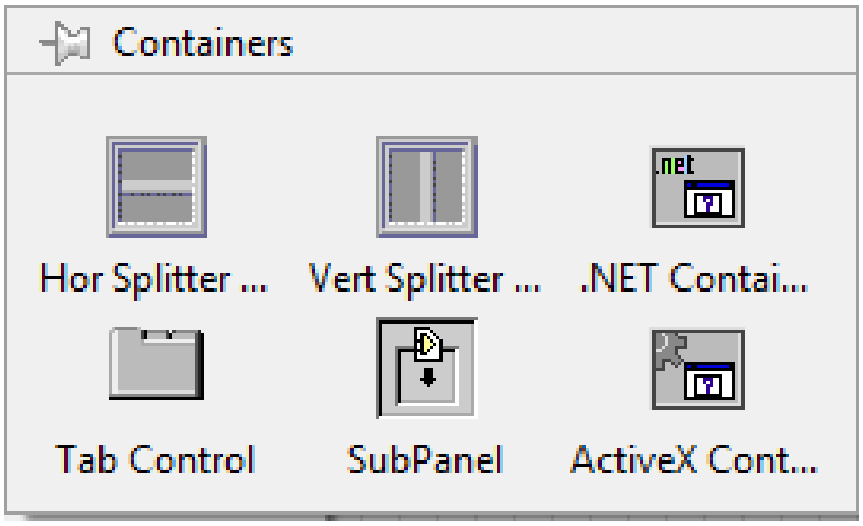
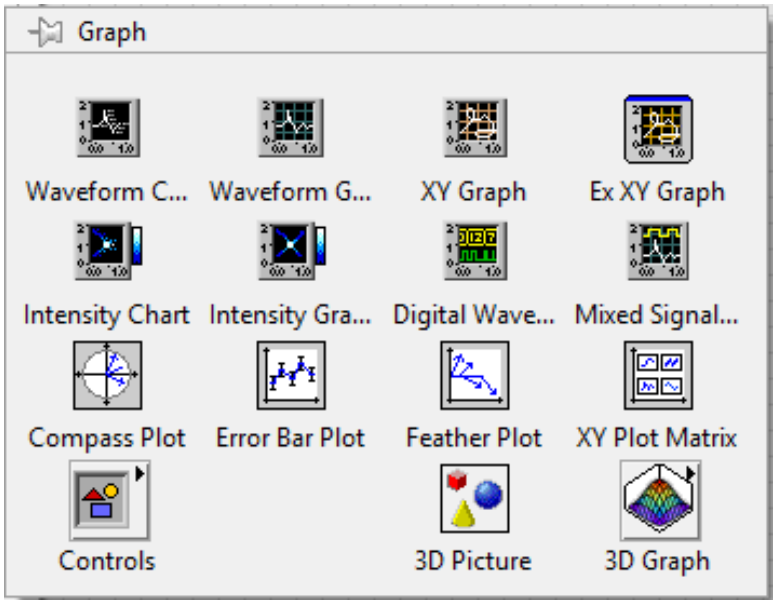


FIG 2.15 Front panel palette

Following are some palettes which are regularly used in making user interface.



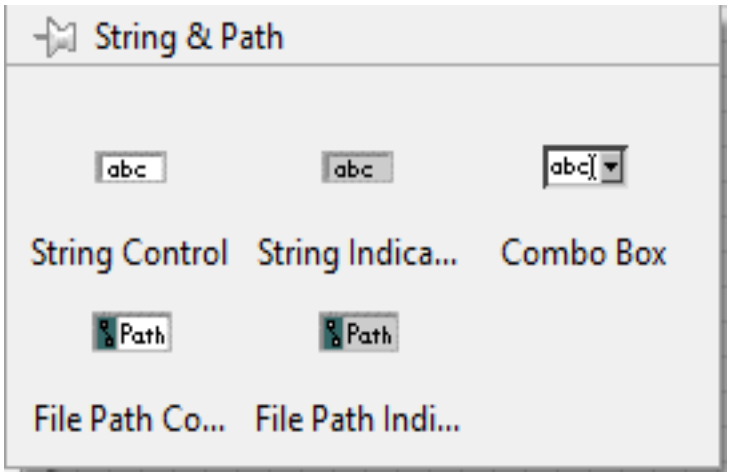


FIG 2.16 Front panel palettes

2.2 Arduino Mega 2560

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board. Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs. Arduino projects can be stand-alone, or they can be communicated with software running on your computer (e.g. Flash, Processing, etc). The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free. The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

Features

- Schematic design of the open source development interface free download, and also according to the needs of their own changes.
- Downloading the program is simple and convenient.
- Simply with the sensor, a wide range of electronic components connection (such as: LED light, buzzer, keypad, photoresistor, etc.), make all sorts of interesting things.
- Using the high-speed micro-processing controller (ATMEGA328).
- The development of language and development environment is very simple, easy to understand, very suitable for beginners to learn.

Arduino is a single-board microcontroller to make using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.

TABLE 2.2 Arduino Mega 2560 specifications

Sr No.	Specifications	Detail
1	Microcontroller	ATmega2560
2	Operating Voltage	5V
3	Input Voltage	7-12V
4	Digital I/O pins	54(14 for PWM o/p)
5	Analog Input pins	16
6	Flash memory	256KB
7	SRAM	8KB
8	EEPROM	4KB
9	Clock Speed	16MHz

Above table shows hardware specifications of Arduino board we have used which is Arduino ATmega2560.

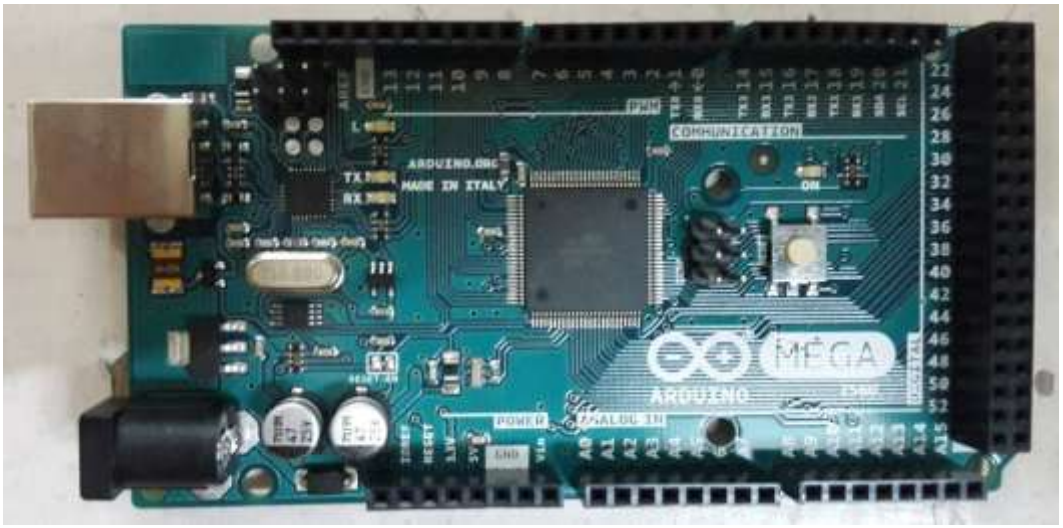
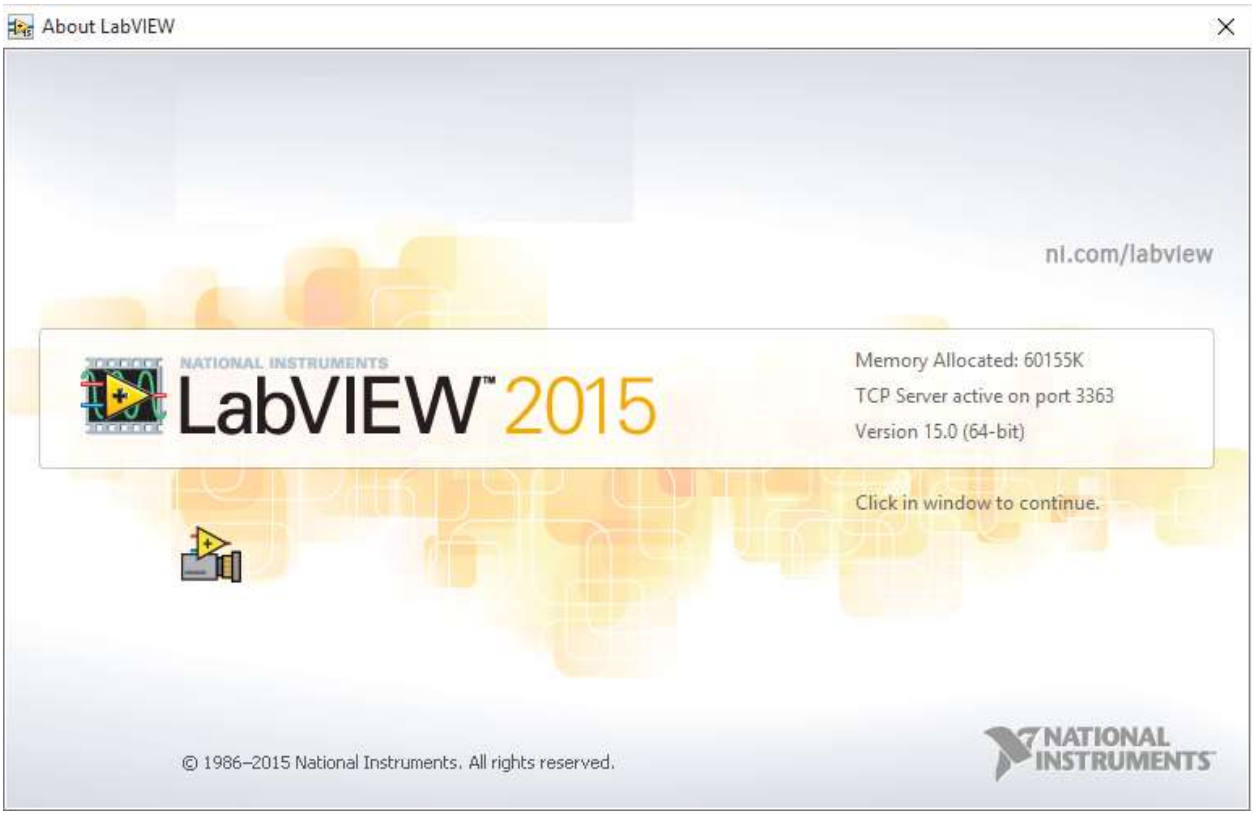


FIG 2.17 Arduino Mega 2560

3 SOFTWARE

3.1 LabView 2015



LabView 2015 is the latest version of labview by NI. It offers new features and additional customization options. The latest version of LabVIEW delivers speed improvements, development shortcuts, and debugging tools to empower developers to efficiently interact with the systems they create.

Why LabView:

Simplify Complexity	Custom User Interface
Data Extensive Analysis	Powerful Multithreaded Execution
Record and Share Measurement	Right Tool, Right Task
Deploy Software to the Right Hardware	

3.1.1 Creating project

Starting from scratch, complete following steps to start with project creation.

- In the Getting Started window, click the Create Project button to display the Create Project dialog box. The Create Project dialog box provides common starting points for LabVIEW projects.
- Select Blank VI from the list of items and click Finish. A blank front panel window and block diagram window appear.
- Display the block diagram.
- If the Functions palette is not visible, right-click any blank space on the block diagram to display a temporary version of the Functions palette. Click the thumbtack, shown below, in the upper left corner of the Functions palette to pin the palette so it is no longer temporary

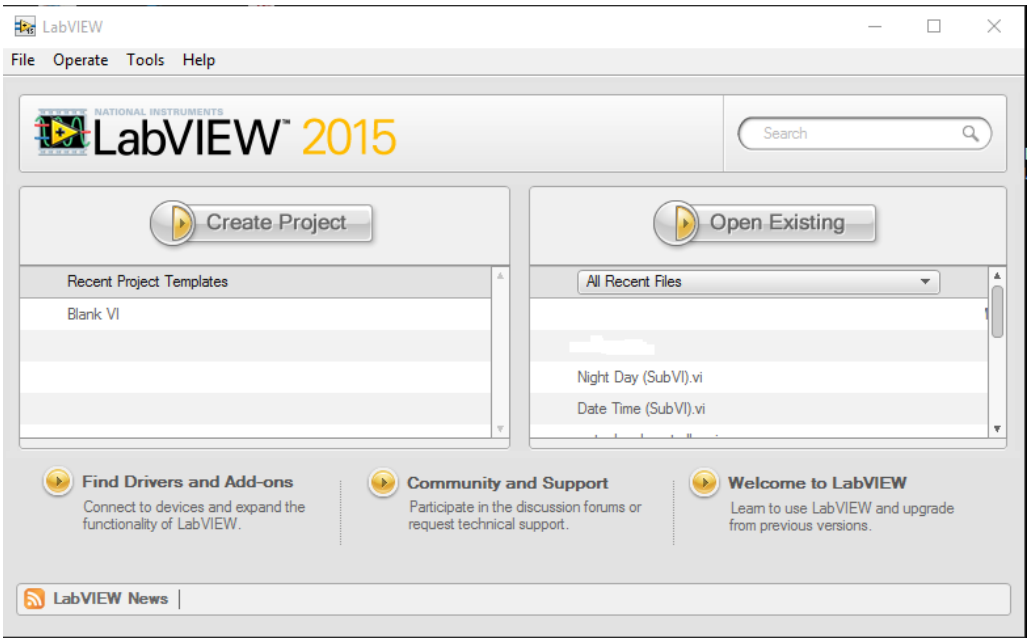


FIG 3.1.1 Create project

If any available templates is suitable then select that template or select blank project or VI.

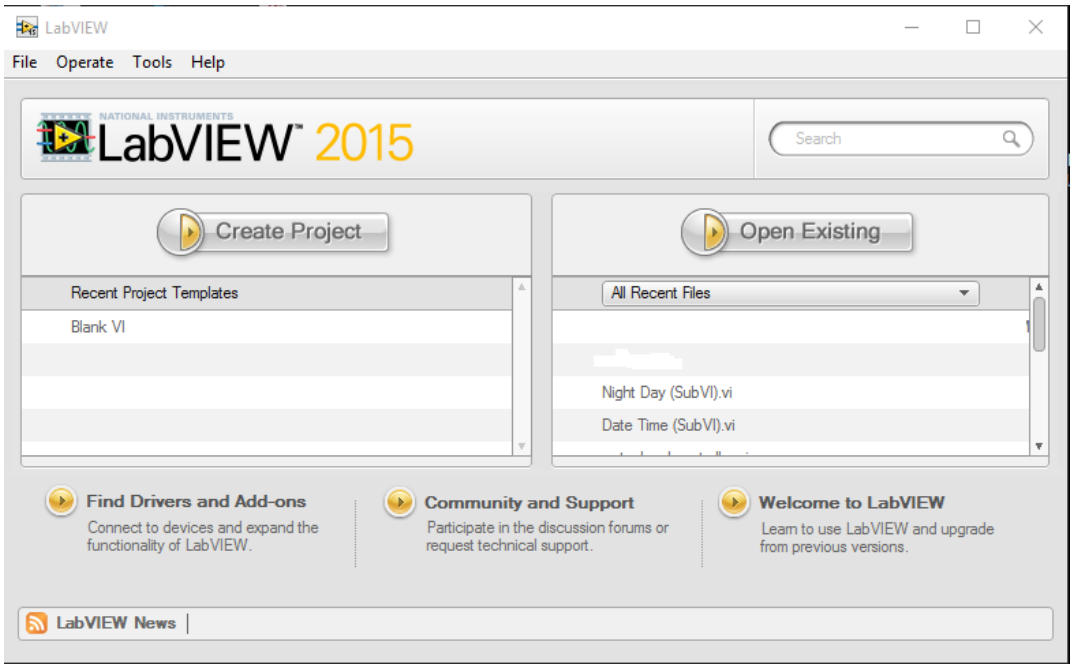


FIG 3.1.2 Create project

Making front panel and block diagrams

As soon as you create a blank project or VI following window will pop up. This window is our working area. The left hand side window is our front panel and right hand side window is block diagram where graphical code is written.

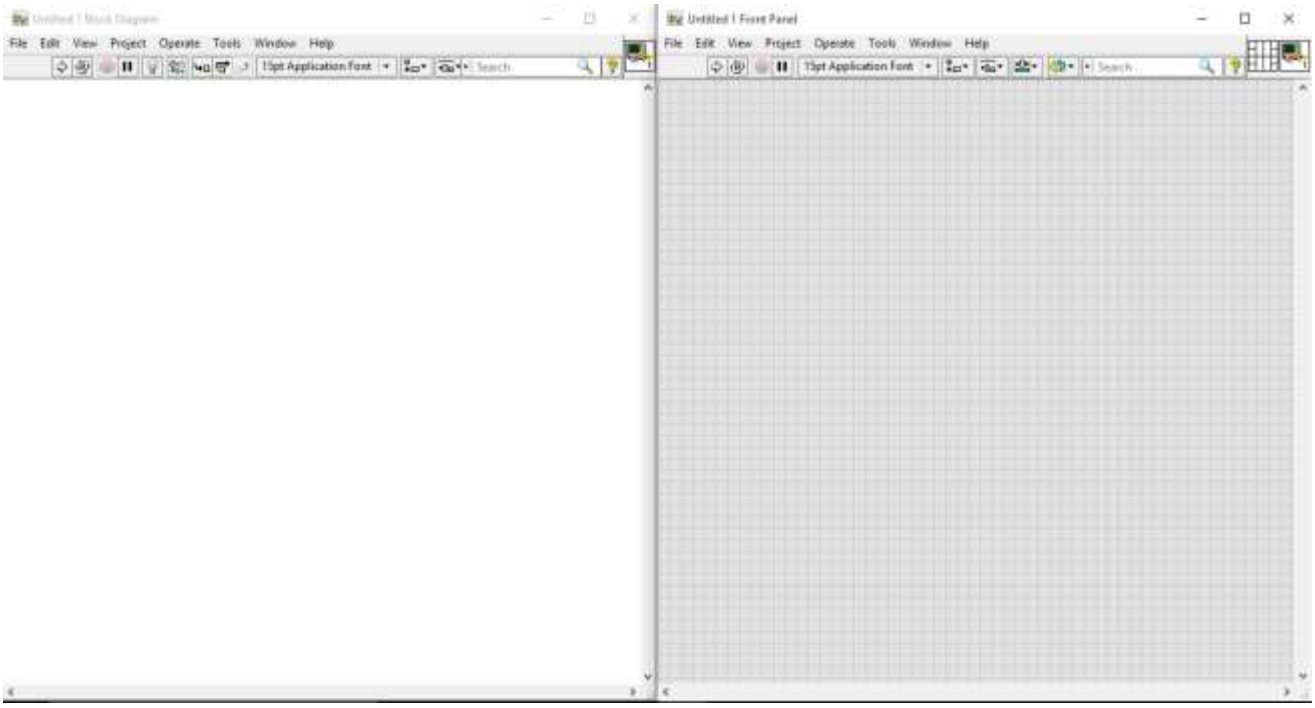


FIG 3.2 front panel and block diagram

As this is done we can start to create our front panel first and build graphical code for the same in block diagram. Making front panel and block diagram can be done easily after LIYTERATURE STUDY of LabView is done.

Running and debugging VI's

To run a VI, you must wire all the subVIs, functions, and structures with the data types the terminals expect. Sometimes a VI produces data or runs in a way you do not expect. You can use LabVIEW to configure how a VI runs and to identify problems with block diagram organization or with the data passing through the block diagram.

Running a VI executes the operation for which you designed the VI. You can run a VI if the Run button on the toolbar appears as a solid white arrow, shown at below. The solid white arrow also indicates you can use the VI as a sub VI if you create a connector pane for the VI. While the VI runs, the Run button changes to darkened arrow, shown at below, to indicate that the VI is running. You cannot edit a VI while the VI runs.

FIG 3.3 Execution and Debugging tools

Fig shows both running and debugging tools of VI. Going from left to right these are as follows:

- Single run
- Run continuous
- Abort execution
- Pause

(execution tools)

- Highlight execution
- Retain wire values
- Step into
- Step over
- Step out

(Single stepping)

(Debugging tools)

After making VI save it and u are good to go.

Apart from standard features available in LabView we have add ons too. Following are the add ons used:

- NI Vision
- LabView interface for Arduino(LIFA)

3.2 Arduino IDE

Arduino programs may be written in any programming language with a compiler that produces binary machine code. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio.

The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such assyntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism to compile and load programs to an Arduino board. A program written with the IDE for Arduino is called a "sketch".

The Arduino IDE supports the languages C and C++ using special rules to organize code. The Arduino IDE supplies a software library called Wiring from the Wiring project, which provides many common input and output procedures

Detailed information regarding Arduino is not needed as it is open software and easy to use.

3.3 LabView Arduino communication using LabVIEW Interface For Arduino

This is the most important section of our project as far as working is concerned. LIFA which stands for “Labview Interface For Arduino” is used to establish communication between Arduino and LabView. LIFA and WEB BASED are the sections we are excited about.

LIFA (LabVIEW Interface For Arduino)

LabVIEW Interface For Arduino is a kind of plug in which must be installed in LabVIEW so that it can recognize and work together with Arduino Microcontroller. For LabVIEW to work together with Arduino, a software called VI package Manager is required which is the recommended method for downloading and managing the LabVIEW add-ons and give an instant access to the add-ons on the LabVIEW tool network, this VI package Manager works with LabVIEW 2009 and later . The following steps are used:

- From VI package Manager browse LabVIEW interface for Arduino
- Install LabVIEW interface for Arduino
- Connect your Arduino to your computer
- Open Arduino IDE
- User this path(for LabVIEW2011 installed on driver C) : C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\LabVIEW Interface for Arduino\Firmware\LIFA_Base
- Open LIFA_Base file and upload it to your Arduino
- Close the Arduino IDE.

After LIFA is added we will see Arduino object in control palette.

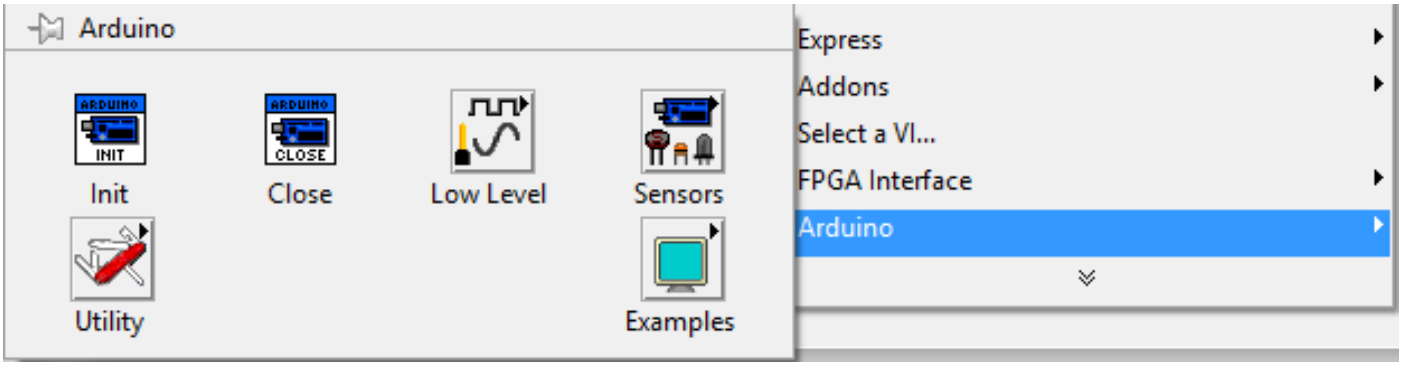


FIG 3.4 LabVIEW Interface For Arduino

3.4 NI Vision

NI Vision for LabVIEW—a part of the NI Vision Development Module—is a library of LabVIEW VIs that you can use to develop machine vision and scientific imaging applications.

NI Vision for LabVIEW is organized into three main function palettes: Vision Utilities, Image Processing, and Machine Vision.

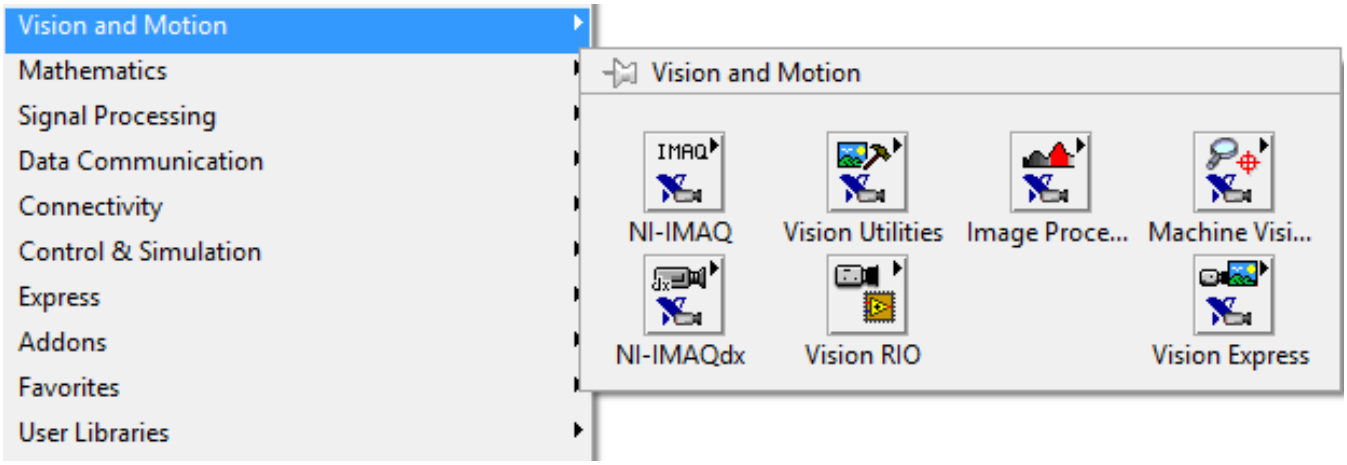


FIG 3.5 Vision and motion

We have used acquire image block from vision and motion to get the image of the intruder.

3.5 Making it WEB BASED

The question that remained till last was to how to make this web based, we have thought of using many things but none gave satisfied or desired results. However LabVIew made it easy for us and is truly a user friendly software. WEB PUBLISHING TOOL of LabView is used to publish the VI online. The computer in which VI has been made acts as a server while others can join in as client. At a time only one computer can operate VI be it server or any of the client.

Web publishing a VI is done as follows:

- Go to Tools»Options»Web Server: Configuration and enable your Web Server.
- Place the name of your VI in Tools»Options»Web Server: Visible Vis
- .Place the network name of the computer that needs to view the VI on Tools»Options»Web Server: Browser Access and select Allow Viewing and Controlling.
- Go to Tools»Web Publishing Tool and follow the instructions on the wizard.

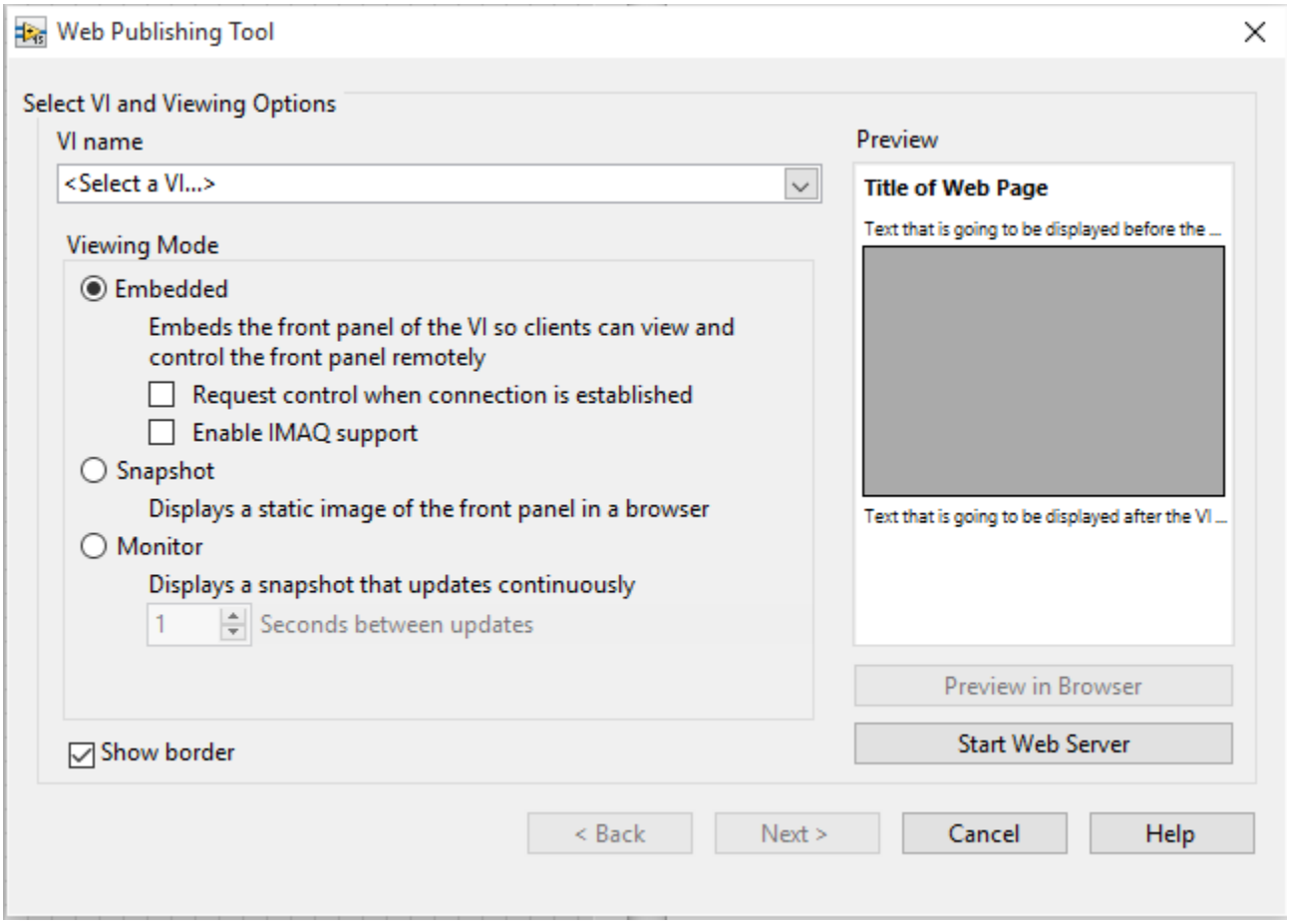


FIG 3.6 Web Publishing Tool

3.6 Making it Network independent

The problem with web publishing tool is that client have to get connected to the same LAN network to which server is connected. So despite of being web based it wasn't feeling like one. Client connected to web must be able to access the VI from anywhere round the world . So we have made use of a commercial app available.

Remote desktop app of google is easily available and easy to use. It can be used in any gadget mobile or laptop. Also it is password protected so there are no issues of security.

We know use of such commercial apps should not be promoted but because of our lack of knowledge in this field we couldn't find other solution.

4 HARDWARE

4.1 Passive Infra-Red (PIR)

The PIR (Passive Infra-Red) Sensor is a pyroelectric device that detects motion by measuring changes in the infrared levels emitted by surrounding objects. This motion can be detected by checking for a high signal on a single I/O pin.

Features:

- Single bit output
- Small size makes it easy to conceal
- Compatible with all Parallax microcontrollers
- 3.3V & 5V operation with <100uA current draw

Theory of Operation

Pyroelectric devices, such as the PIR sensor, have elements made of a crystalline material that generates an electric charge when exposed to infrared radiation. The changes in the amount of infrared striking the element change the voltages generated, which are measured by an on-board amplifier.

The device contains a special filter called a Fresnel lens, which focuses the infrared signals onto the element. As the ambient infrared signals change rapidly, the on-board amplifier trips the output to indicate motion.

Top View

Bottom View

FIG 4.1 PIR

TABLE 4.1 PIR pin specification

Sr No.	Pin	Name	Function
1	-	GND	Connects to Ground or Vss
2	+	Vcc	Connects to Vdd (3.3V to 5V) @ ~100uA
3	OUT	Output	Connects to an I/O pin set to INPUT mode

Calibration

The PIR Sensor requires a 'warm-up' time in order to function properly. This is due to the settling time involved in 'learning' its environment. This could be anywhere from 10-60 seconds. During this time there should be as little motion as possible in the sensors field of view.

Sensitivity

The PIR Sensor has a range of approximately 20 feet. This can vary with environmental conditions. The sensor is designed to adjust to slowly changing conditions that would happen normally as the day progresses and the environmental conditions change, but responds by making its output high when sudden changes occur, such as when there is motion.

4.2 LM35

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The sensor circuitry is sealed and therefore it is not subjected to oxidation and other processes. With **LM35**, temperature can be measured more accurately than with a thermistor.

Features:

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- Rated for full −55° to +150°C range
- Operates from 4 to 30 volts
- Low impedance output, 0.1 W for 1 mA load

FIG 4.2 LM35

TABLE 4.2 LM35 pin specification

Sr No.	Pin No	Name	Function
1	1	Vcc	Supply voltage; 5V (+35V to -2V)
2	2	Output	Output voltage (+6V to -1V)
3	3	Ground	Ground (0V)

4.3 L298N MOTOR DRIVER

The L298 is an integrated monolithic circuit in a 15-lead Multi watt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

Features:

- Operating supply voltage upto 46V
- Total DC current upto 4A
- Low saturation voltage
- Over temperature protection
- High noise immunity (logical "0" input voltage upto 1.5V)

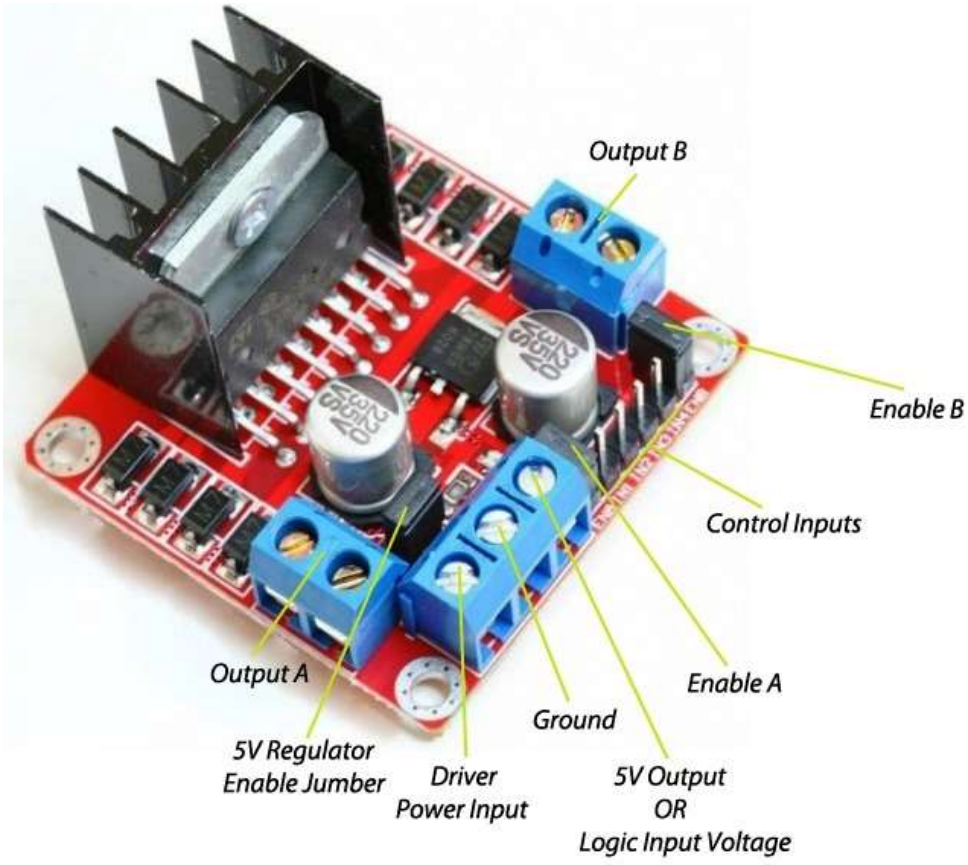


FIG 4.3 Motor driver L298N

4.4 MQ135 AIR QUALITY SENSOR

Sensitive material of MQ135 gas sensor is SnO2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Change of conductivity corresponds to output signal of gas concentration. MQ135 gas sensor has high sensitity to Ammonia, Sulfide and Benze steam, also sensitive to smoke and other harmful gases. It is with low cost and suitable for different application.

Features:

- Good sensitivity to Harmful gases in wide range
- High sensitivity to ammonia, sulfide and benze
- Long life and low cost
- Simple drive circuit

FIG 4.4 MQ135

NOTE: MQ135 gives "0" i.e 0V output on detection of smoke while in normal condition it gives output 1 i.e 5V.

4.5 RELAY MODULE

8 Channel Relay Board is a simple and convenient way to interface 8 relays for switching application in your project. Input voltage level support TTL as well as CMOS. Easy interface with Micro controllers based projects and analog circuits. but in this bord is old . but PCB of new version of 8 channel relay module. with handle AC supply directly for home automation. this module are work same as our home distribution panel board.

FIG 4.5 Relay module

Specifications:

Input AC supply for switching application like Fan, Motor, etc.

Input supply 12 VDC @ 336 mA

Output eight SPDT relay

Relay specification 5 A @ 230 VAC

Trigger level 2 ~ 15 VDC

Header connector for connecting power and trigger voltage

LED on each channel indicates relay status

Screw terminal connector for easy relay output and aux power connection

Four mounting holes of 3.2 mm each

NOTE: Our relay module has reverse logic. So when we give 0v as control input relay gets connected.

4.6 DC MOTOR

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

FIG 4.6 DC motor

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills.

4.7 ARDUINO

An Arduino is an open-source microcontroller development board. In plain English, you can use the Arduino to read sensors and control things like motors and lights. This allows you to upload programs to this board which can then interact with things in the real world. With this, you can make devices which respond and react to the world at large.

For instance, you can read a humidity sensor connected to a potted plant and turn on an automatic watering system if it gets too dry. Or, you can make a stand-alone chat server which is plugged into your internet router. Or, you can have it tweet every time your cat passes through a pet door. Or, you can have it start a pot of coffee when your alarm goes off in the morning.

Basically, if there is something that is in any way controlled by electricity, the Arduino can interface with it in some manner. And even if it is not controlled by electricity, you can probably still use things which are (like motors and electromagnets), to interface with it.

The possibilities of the Arduino are almost limitless. As such, there is no way that one single tutorial can cover everything you might ever need to know. That said, I've done my best to give a basic overview of the fundamental skills and knowledge that you need to get your Arduino up and running. If nothing more, this should function as a springboard into further experimentation and learning.

Different types of Arduinos

- Arduino Uno
- Arduino NG, Diecimila, and the Duemilanove (Legacy Versions)
- Arduino Mega 2560
- Arduino Mega ADK
- Arduino LilyPad

Arduino Mega 2560

The Mega is the second most commonly encountered version of the Arduino family. The Arduino Mega is like the Arduino Uno's beefier older brother. It boasts 256 KB of memory (8 times more than the Uno). It also had 54 input and output pins, 16 of which are analog pins, and 14 of which can do PWM. However, all of the added functionality comes at the cost of a slightly larger circuit board. It may make your project more powerful, but it will also make your project larger.

TABLE 4.7 Arduino atmega2560

Sr No.	Specifications	Detail
1	Microcontroller	ATmega2560
2	Operating Voltage	5V
3	Input Voltage	7-12V
4	Digital I/O pins	54(14 for PWM o/p)
5	Analog Input pins	16
6	Flash memory	256KB
7	SRAM	8KB
8	EEPROM	4KB
9	Clock Speed	16MHz

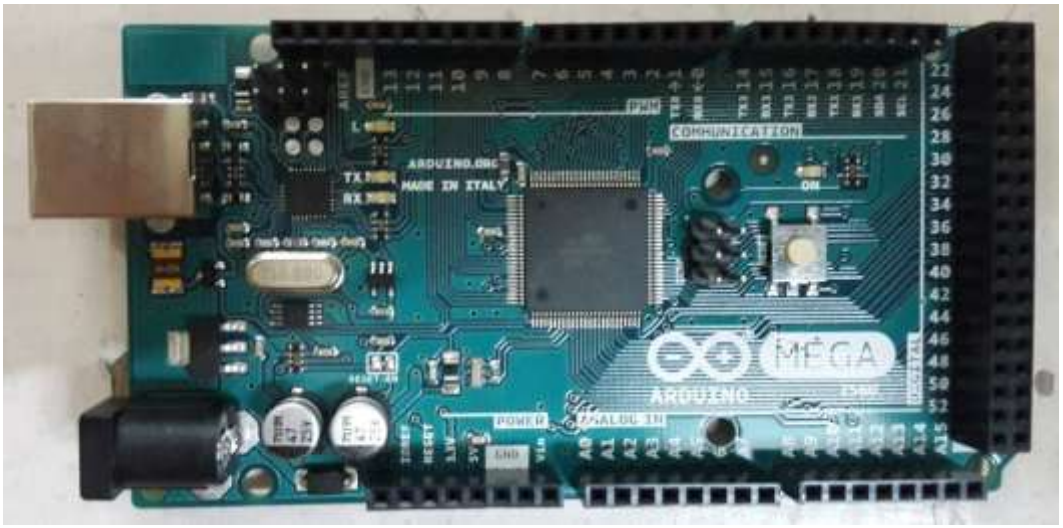


FIG 4.7 Arduino mega

4.8 16X2 LCD

LCD (Liquid Crystal Display) screen is an electronic display module and find a wide range of applications. A 16x2 LCD display is very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi segment LEDs. The reasons being: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations and so on.

A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. This LCD has two registers, namely, Command and Data.

The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. Click to learn more about internal structure of a LCD.

FIG 4.8 LCD

4.9 4x4 Matrix Membrane Keypad

Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically an arduino. Underneath each key is a pushbutton, with one end connected to one row, and the other end connected to one column.

In order for the arduino to determine which button is pressed, it first needs to pull each of the four columns (pins 1-4) either low or high one at a time, and then poll the states of the four rows (pins 5-8). Depending on the states of the columns, the arduino can tell which button is pressed.

Features

- Ultra-thin design
- Adhesive backing
- Excellent price/performance ratio

Key Specifications

- Maximum Rating: 24 VDC, 30 mA
- Interface: 8-pin access to 4x4 matrix
- Operating temperature: 32 to 122 °F (0 to 50°C)
- Dimensions:
  Keypad, 2.7 x 3.0 in (6.9 x 7.6 cm)
  Cable: 0.78 x 3.5 in (2.0 x 8.8 cm)

Application Ideas

- Security systems
- Menu selection
- Data entry for embedded systems

FIG 4.9 keypad

5 PROGRAMMING AND IMPLEMENTATION

5.1 Pin assignment

		Input to Arduino			Output from arduino		
Tag Name	**Component Type**	**Input Pin Type**	**Pin No.**	**Description**	**Output Pin Type**	**Pin No.**	**Description**
H-PR-1	INPUT	DIGITAL	22	PIR OUT => PIN 22			
H-LT	OUTPUT				DIGITAL	23	PIN 23 => RLY-BD-1 IN1
H-PR-2	INPUT	DIGITAL	24	PIR OUT => PIN 24			
H-LT	OUTPUT				DIGITAL	25	PIN 25 => RLY-BD-1 IN2
H-LM35	INPUT	ANALOG	A0	LM-35 CENTER PIN => PIN A0			
H-FAN	OUTPUT				PWM (ANALOG)	2	PIN 2 => MTR-DR-1 ENA
LS-OUT	INPUT	DIGITAL	26	LMT SW CENTER PIN => PIN 26			
LS-IN	INPUT	DIGITAL	27	LMT SW CENTER PIN => PIN 27			
LS-MTR	OUTPUT				PWM (ANALOG)	5	PIN 5 => MTR-DR-1 ENB
					DIGITAL	29	PIN 29 => MTR-DR-1 IN3
					DIGITAL	30	PIN 30 => MTR-DR-1 IN4
K-PR	INPUT	DIGITAL	31	PIR OUT => PIN 31			

K-LT	OUTPUT				DIGITAL	32	PIN 32 => RLY-BD-1 IN3
K-MQ	INPUT	DIGITAL	33	MQ D-OUT => PIN 33			
K-BZZR	OUTPUT				PWM (ANALOG)	3	PIN 3 => BUZZER VCC(+) PIN
B-PR	INPUT	DIGITAL	34	PIR OUT => PIN 34			
B-LT	OUTPUT				DIGITAL	35	PIN 35 => RLY-BD-1 IN4
B-LM35	INPUT	ANALOG	A1	LM-35 CENTER PIN => PIN A1			
	OUTPUT				PWM (ANALOG)	4	PIN 4 => MTR-DR-2 ENB
W-PR	INPUT	DIGITAL	36	PIR OUT => PIN 36			
W-LT	OUTPUT				DIGITAL	37	PIN 37 => RLY-BD-1 IN5
W-SW	INPUT	DIGITAL	38	TGL SW CENTER PIN => PIN 38			
P-PR	INPUT	DIGITAL	39	PIR OUT => PIN 39			
P-LT	OUTPUT				DIGITAL	40	PIN 40 => RLY-BD-1 IN6
	INPUT	DIGITAL	41	LMT SW CENTER PIN => PIN 41			
	INPUT	DIGITAL	42	LMT SW CENTER PIN => PIN 42			
	OUTPUT				DIGITAL	43	PIN 43 => MTR-DR-2 ENA
					DIGITAL	44	PIN 44 => MTR-DR-2 IN1
					DIGITAL	45	PIN 45 => MTR-DR-2 IN2

PASS-1	INPUT	DIGITAL	46	MEGA2 PIN 30 => PIN 46			
PASS-2		DIGITAL	47	MEGA2 PIN 31 => PIN 47			
MAIL	INPUT	DIGITAL	48	LMT SW CENTER PIN => PIN 48			
T-LOW	INPUT	ANALOG	A2	T-LOW => PIN A2			
T-UP	INPUT	ANALOG	A3	T-LOW => PIN A3			
T-PMP	OUTPUT				DIGITAL	49	PIN 49 => RLY-BD1 IN7

TABLE 5.1 Arduino pin assignment

5.2 VI's

LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: a block diagram, a front panel and a connector panel. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators.

Final VI having program of entire house Is big and complicated and so we are going from small to big.

5.2.1 Sub VI's

Just like "function" in C programing we have Sub VI in LabView. Logics which are often used in program are made in a sub VI and this sub VI is then used everywhaere.

Digital Read VI

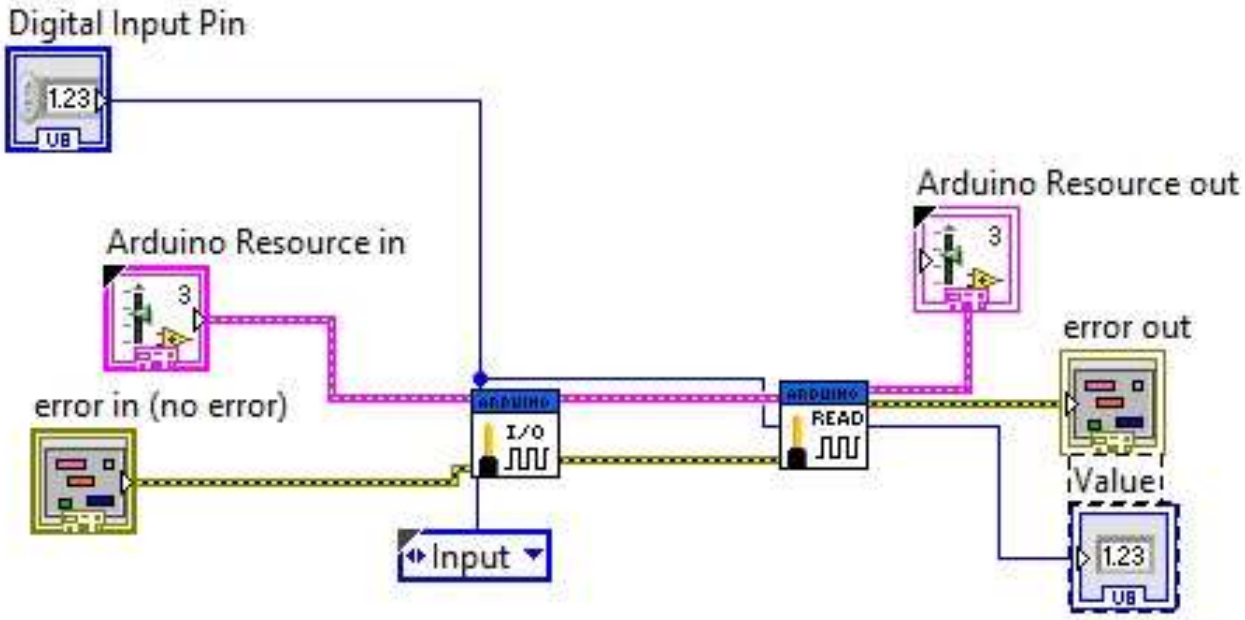
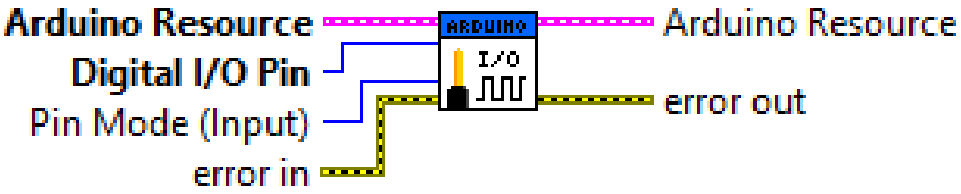


FIG 5.1 Digital Read sub VI

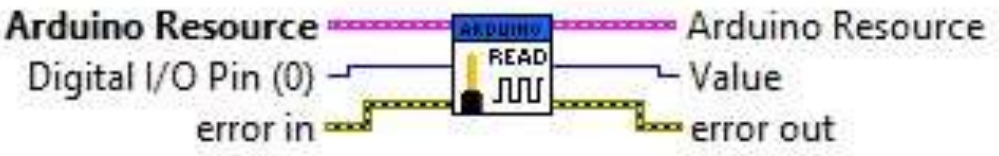
- **Set Digital Pin Mode**

  Configure the specified digital I/O pin as either input or output.



- **Digital Read Pin**

  Read the digital value of the selected Arduino digital input pin. The pin must be configured first as an input using Set Digital Pin Mode VI.



Digital Write VI

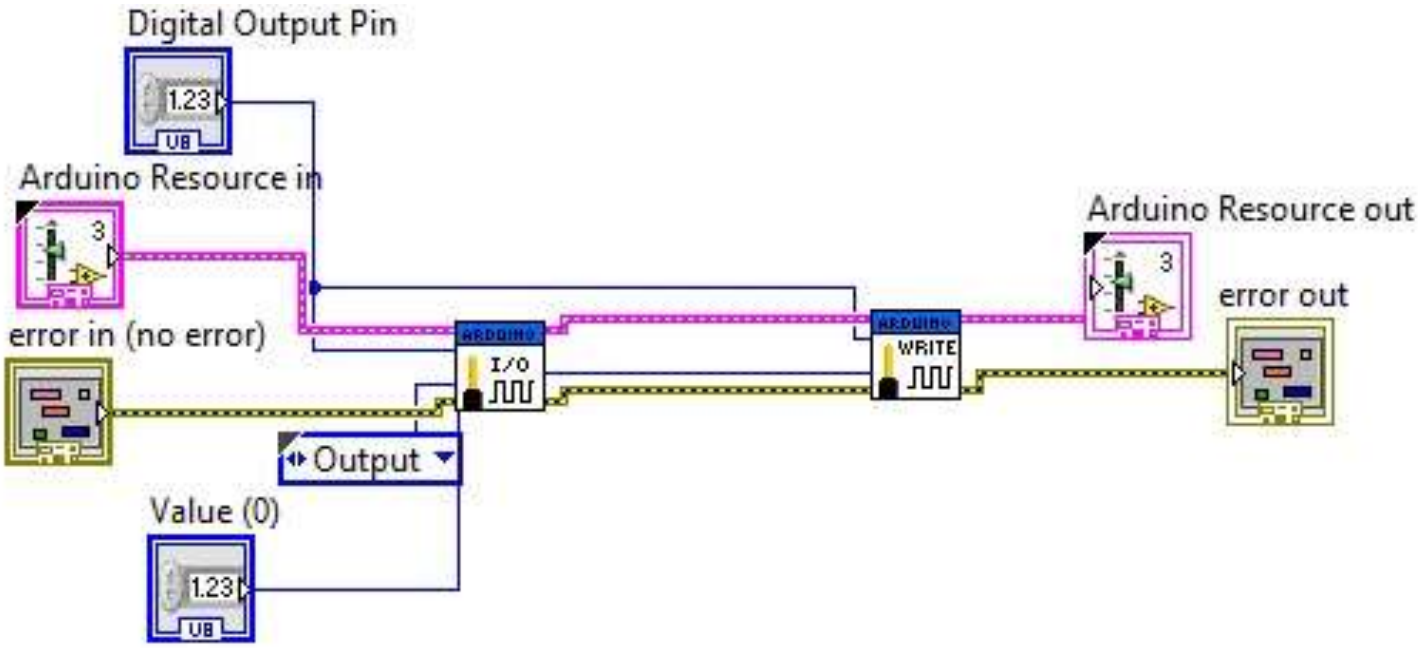
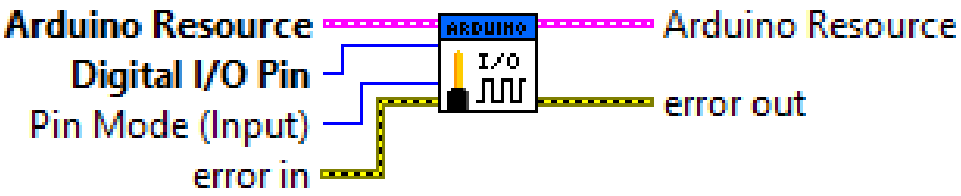


FIG 5.2 Digital write sub VI

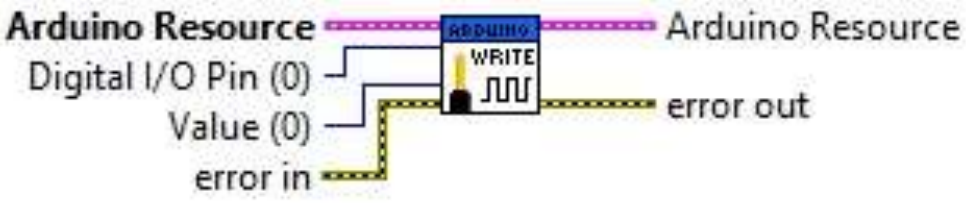
- **Set Digital Pin Mode**

  Configure the specified digital I/O pin as either input or output.



- **Digital Write Pin**

Write the specified value on the selected Arduino digital output pin. The pin must be configured first as an ouput using Set Digital Pin Mode VI.



Integer Not Gate

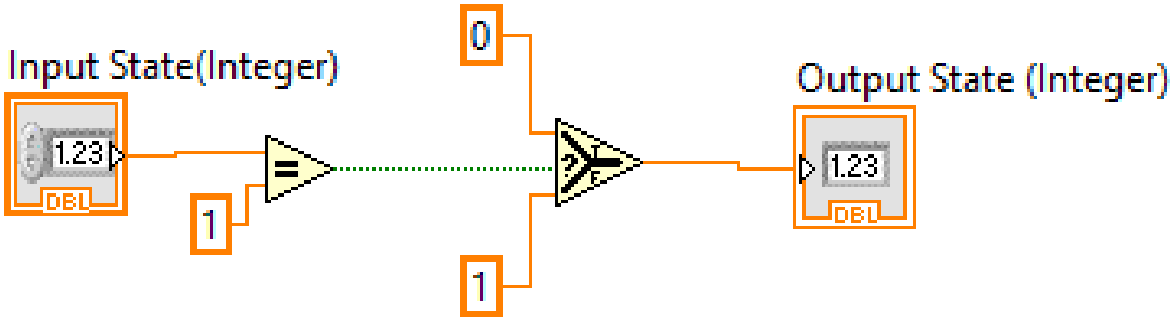
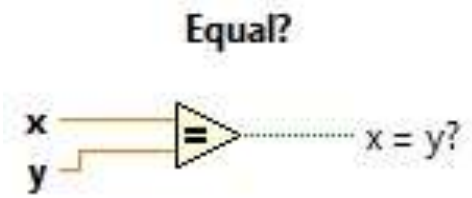


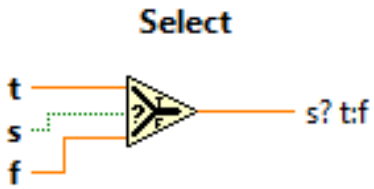
FIG 5.3 Integer not gate sub VI

- **Equal**
  Returns TRUE if **x** is equal to **y**. Otherwise, this function returns FALSE



- **Select**

  Returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**.



Night Day

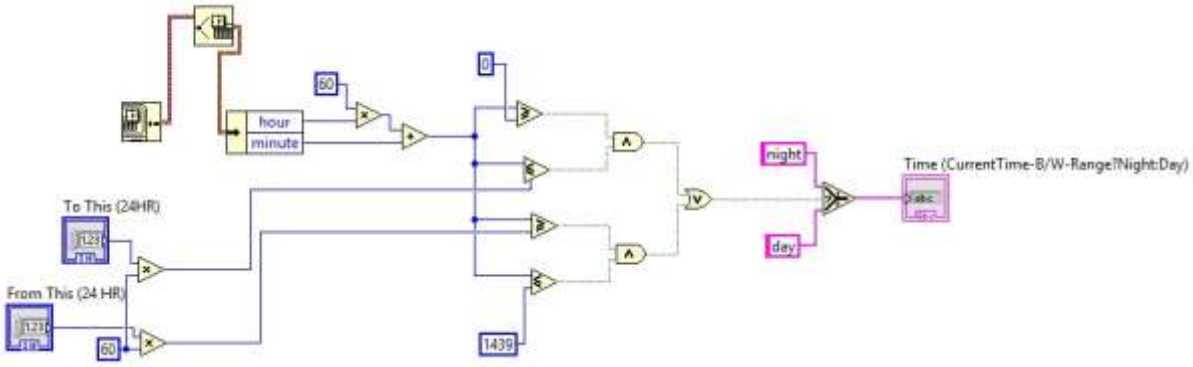
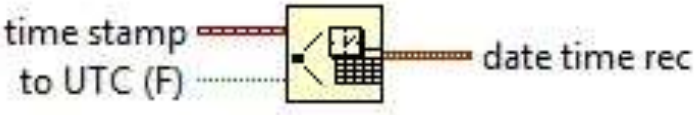


FIG 5.4 Night and day sub VI (See Appendix Fig-B)

- **Get Date/Time In Seconds**
  Returns a timestamp of the current time. LabVIEW calculates this timestamp using the number of seconds elapsed since 12:00 a.m., Friday, January 1, 1904, Universal Time.

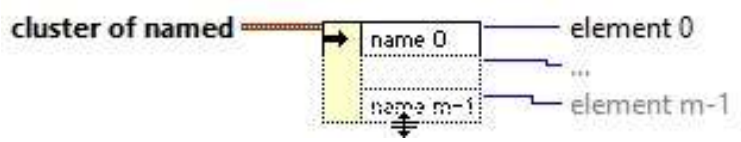
- **Seconds To Date/Time Function**

  Converts a timestamp value or a numeric value to a cluster of time values. This function loses fractional seconds of precision when converting the timestamp. If you convert **date time rec** back into a time stamp, the timestamp may not display the exactly correct value.



- **Unbundle By Name Function**

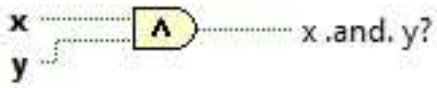
  Returns the cluster elements whose names you specify. You do not have to keep track of the order of the elements within the cluster. This function does not require



the number of elements to match the number in the cluster. After you wire a cluster to this function, you can select an individual element from the function.

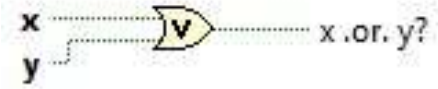
- **And Function**

  Computes the logical AND of the inputs. Both inputs must be Boolean values, numeric values, or error clusters. If both inputs are TRUE, the function returns TRUE. Otherwise, it returns FALSE.



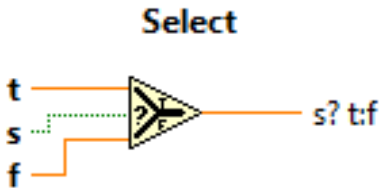
- **Or Function**

  Computes the logical OR of the inputs. Both inputs must be Boolean values, numeric values, or error clusters. If both inputs are FALSE, the function returns FALSE. Otherwise, it returns TRUE.



- **Select**

  Returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**.



Explanation

Night day Sub VI is used for selecting the time during which the VI should work. In Night day Sub VI first of all the current time is converted into Sec by Date/Time into Sec Block and then Sec is converted into date and time by Seconds To Date/Time

Function. This converts a timestamp value or a numeric value to a cluster of time values. Unbundle by Name Block will returns the cluster elements whose names you specify. The current hours will be then multiplied by 60 and added to the minute with the help of Add function. This minute is compared with 0 by greater or equal function if time is greater or equal to 0 then it will return 1. During 18:00 to 6:00 am this VI will be in automatic mode and during day time this VI will run only in manual mode this has been done to use natural sunlight during day time in order to it energy efficient. Let us take an example if the current time is 10 am that is 600 min which is greater than 0 and this min is greater than to the time which we have taken 360 min so the input to 1st AND gate will be 1,0 respectively so the output of 1st AND gate will be 0. Now this time is compared for less than equal to with 1439 min so output will be 1. In other comparision 600 min is less than 1080 min so it output will be 0. So input to 2nd AND gate will be 1,0 respectively so it output will be 0. Now the output of both AND gate are given as input to OR gate so output will be 0 so Select block will select day. If the output of OR gate is 1 the condition will be true and Select block will select night.

Date Time

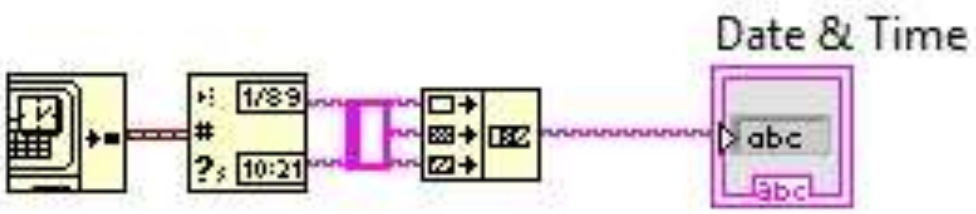


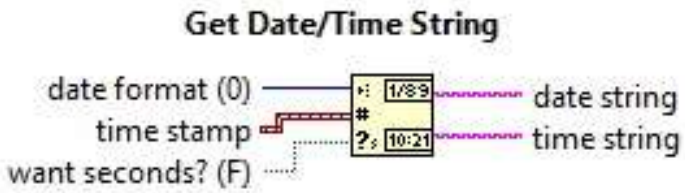
FIG 5.5 Date time sub VI

- **Get Date/Time In Seconds**

  Returns a timestamp of the current time. LabVIEW calculates this timestamp using the number of seconds elapsed since 12:00 a.m., Friday, January 1, 1904, Universal Time.

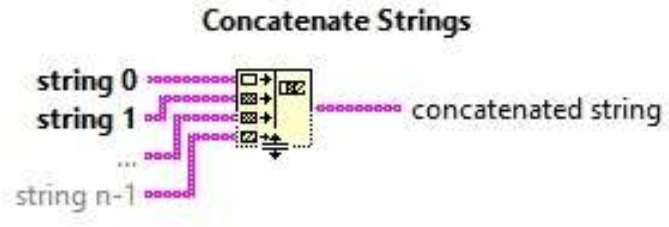
- **Get Date/Time String Function**

  Converts a timestamp value or a numeric value to a date and time string in the time zone configured for the computer. The function interprets timestamp and numeric values as the time-zone-independent number of seconds that have elapsed since 12:00 a.m., Friday, January 1, 1904, Universal Time.



- **Concatenate Strings Function**

  Concatenates input strings and 1D arrays of strings into a single output string. For array inputs, this function concatenates each element of the array.



Password Condition

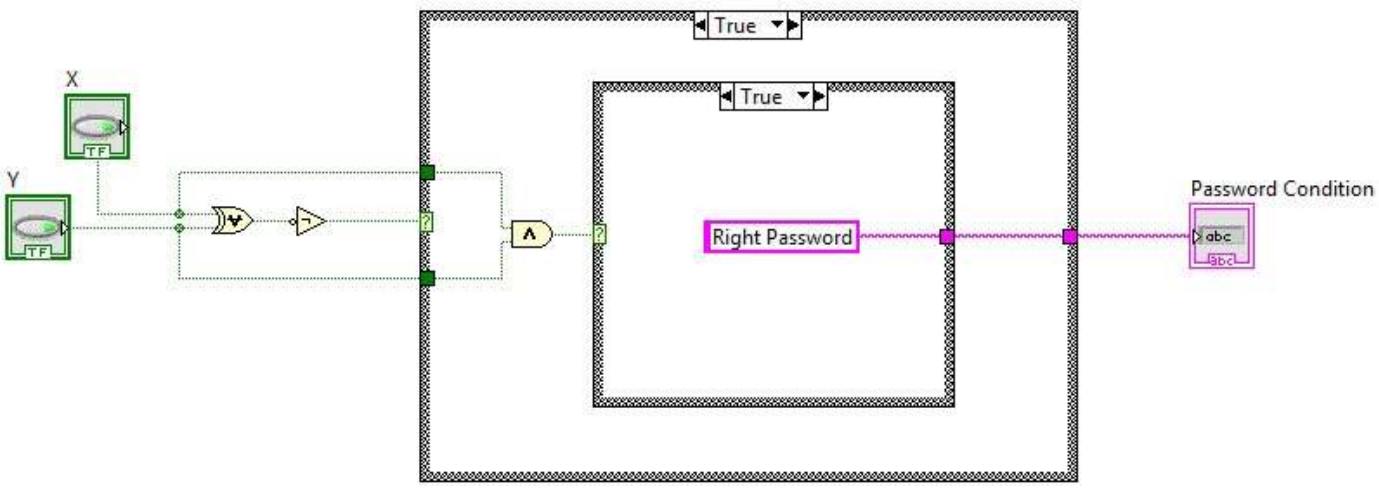
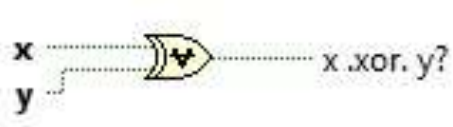


FIG 5.6 Password condition sub VI (See Appendix Fig-C)

- **Exclusive Or Function**

  Computes the logical exclusive or (XOR) of the inputs. Both inputs must be Boolean values, numeric values, or error clusters. If both inputs are TRUE or both inputs are FALSE, the function returns FALSE. Otherwise, it returns TRUE.

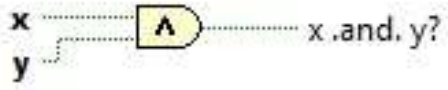


- **Not Function**

  Computes the logical negation of the input. If **x** is FALSE, the function return TRUE. If **x** is TRUE, the function returns FALSE.

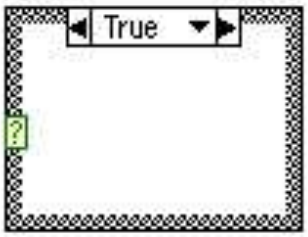
- **And Function**

Computes the logical AND of the inputs. Both inputs must be Boolean values, numeric values, or error clusters. If both inputs are TRUE, the function returns TRUE. Otherwise, it returns FALSE.



- **Case Structure**

  Contains one or more subdiagrams, or cases, exactly one of which executes when the structure executes. The value wired to the case selector determines which case to execute.



Explanation

When X and Y which are input are taken as 1 respectively their XOR will give FALSE condition then negation of the input are taken which will enter case structure if the condition is true otherwise it will enter false case structure and it will display password condition is in normal condition . when X and Y are 1 their AND will give output true and password condition will be right password.

PATH

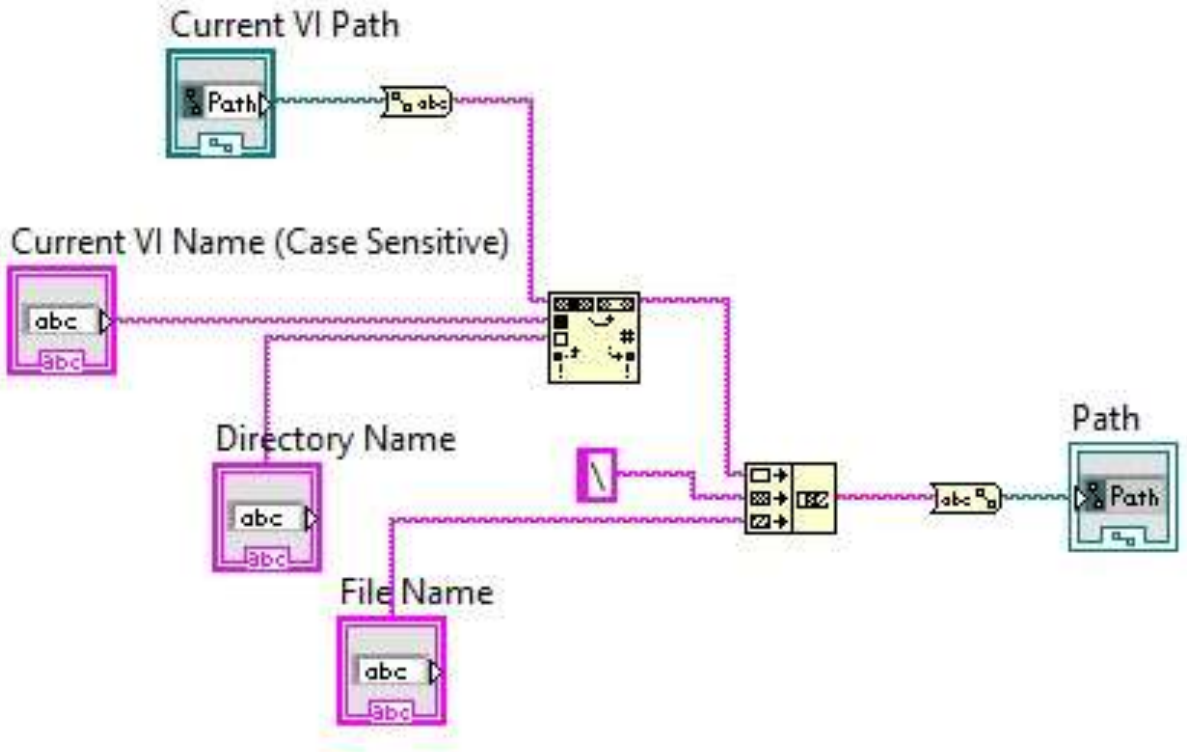
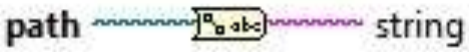


FIG 5.7 Path sub VI

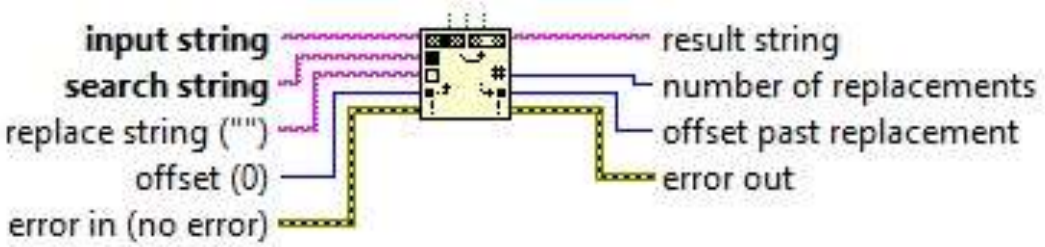
- **Path To String Function**

  Converts path into a string describing a path in the standard format of the platform.



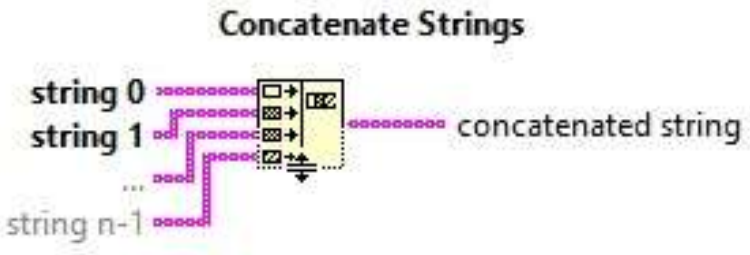
- **Search and Replace String Function**

  Replaces one or all instances of a substring with another substring. To include the multiline input and enable advanced regular expression searches, right-click the function and select Regular Expression.

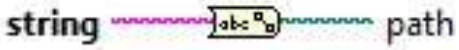


- **Concatenate Strings Function**

Concatenates input strings and 1D arrays of strings into a single output string. For array inputs, this function concatenates each element of the array.



- **String To Path Function**
  Converts a string, describing a path in the standard format for the current platform, to a path.



Email

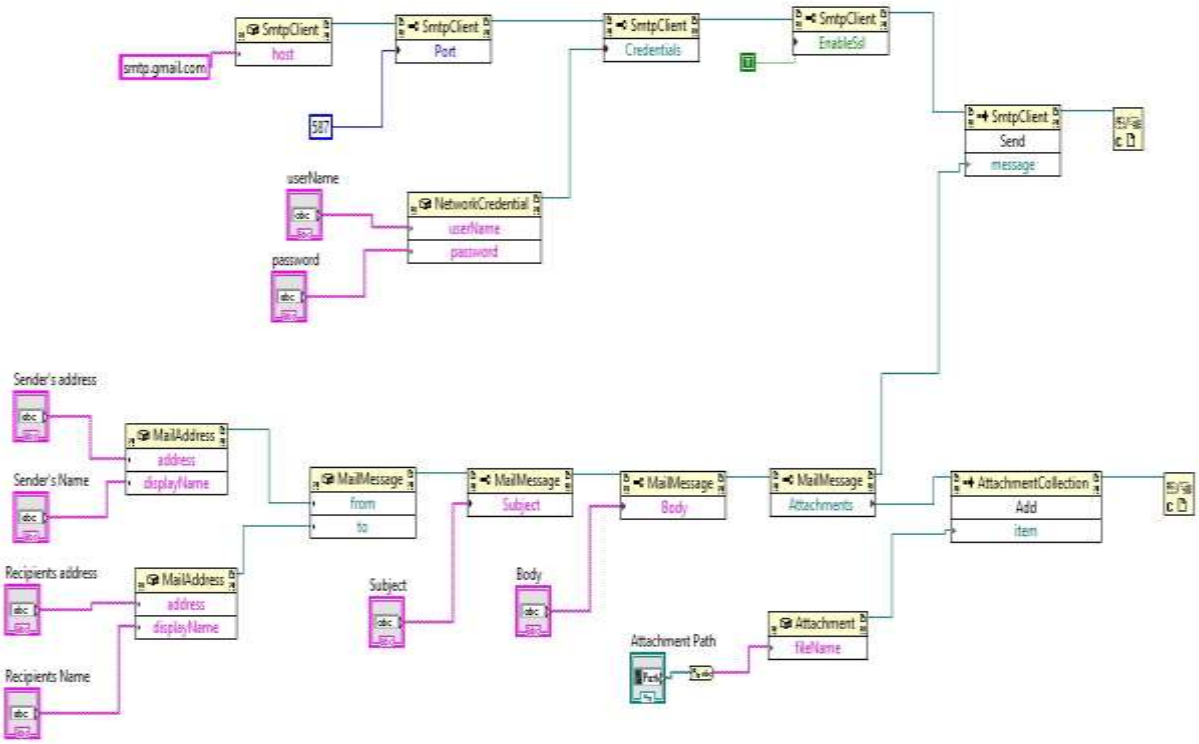
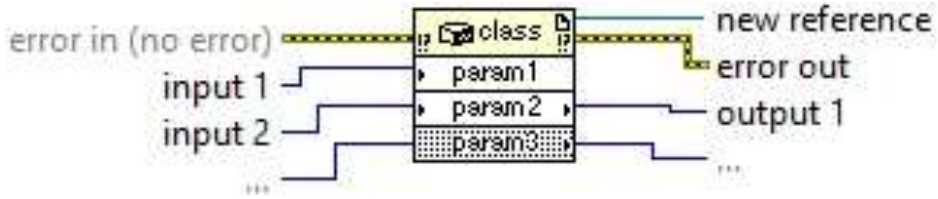


FIG 5.8 Email sub VI (See Appendix Fig-D)

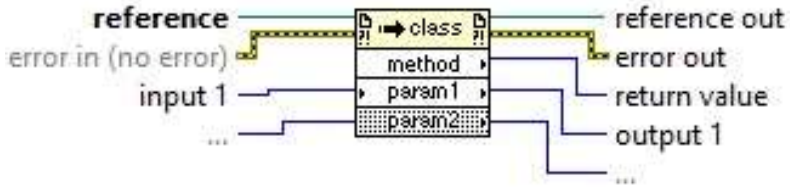
- **Constructor Node**

  Creates an instance of a .NET object. This node identifies the constructor from which to create a .NET object.



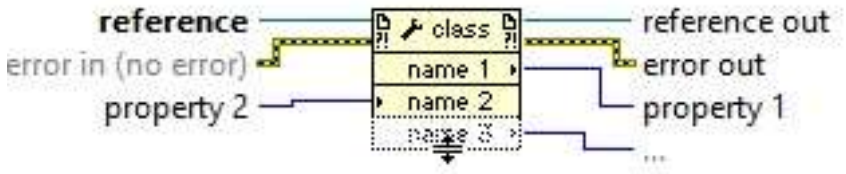
- **Property Node**

  Gets (reads) and/or sets (writes) properties of a reference. Use the property node to get or set properties and methods on local or remote application instances, Vis, and objects. You also can use the property Node to access the private data of a LABVIEW class.



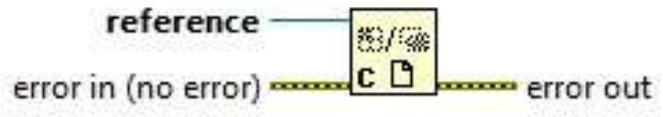
- **Invoke Node**

  Invokes a method or action on a reference. Most methods have associated parameters.



- **Close Reference Function**

  Closes a refnum associated with an open VI, VI object, an open application instance, or a .NET or ActiveX object.



5.2.2 HALL VI

VI of hall uses digital read, digital write, day and night and integer not gate sub VI. It has two modes of operation which are:

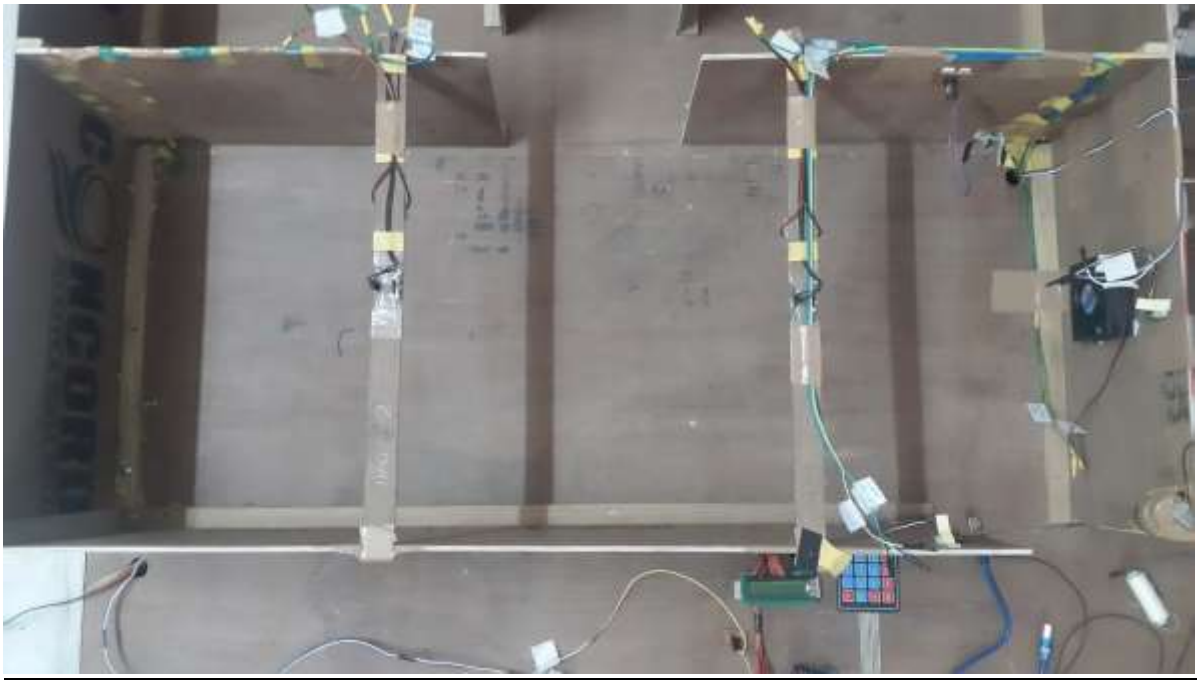
- Manual
- Automatic

Case structure is used for switching between manual and automatic modes.

TABLE 5.2.2 Hall input output

Sr No.	Digital input	Digital output
1	PIR	Light
2	Keyboard	Mail(with pic)
3	Limit Switch	Door motor

Sr No.	Analog Input	Analog Output
1	LM 35	Fan



Manual

We can operate entire functions of hall manually via web.

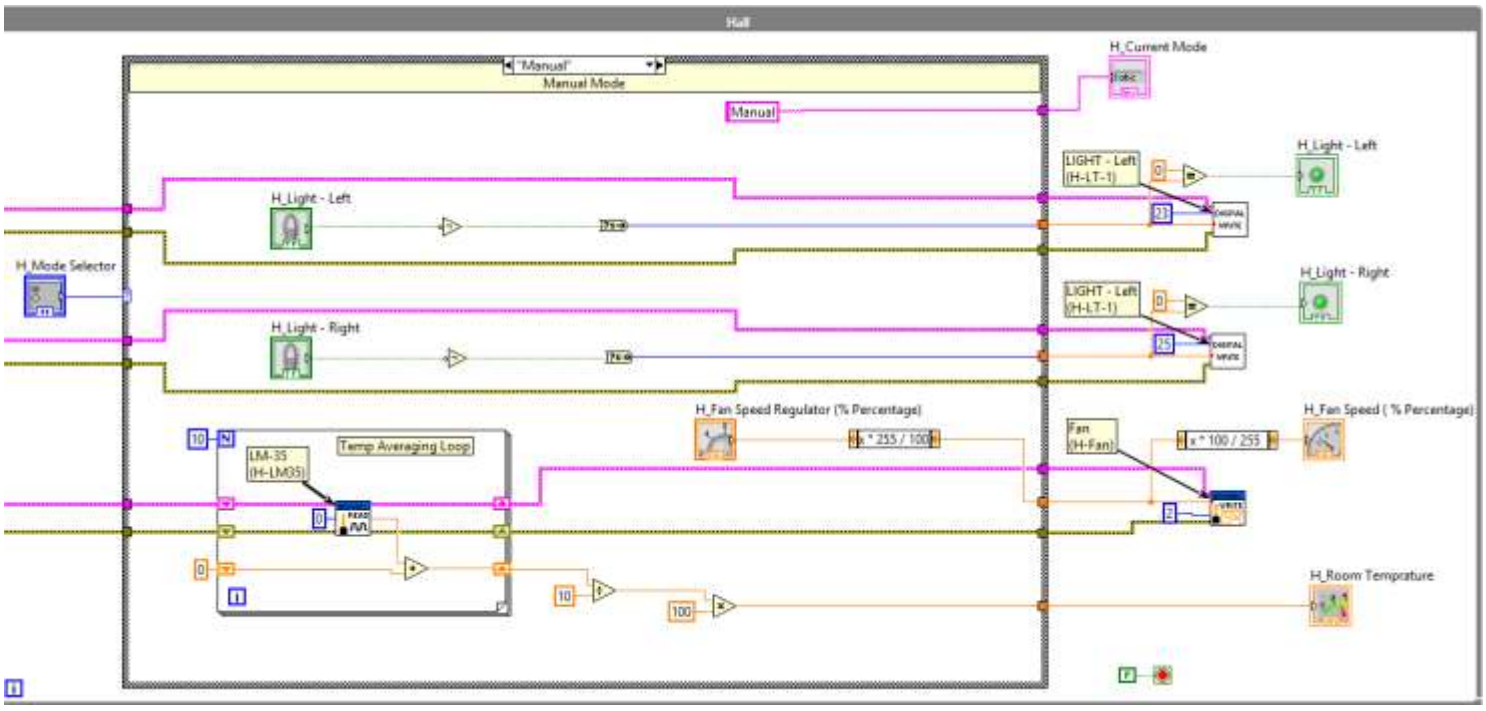


FIG 5.9 Hall manual block diagram (See Appendix Fig-E)

Client won't see above window nor will he be able to change the VI. Client will see following front panel on his screen.

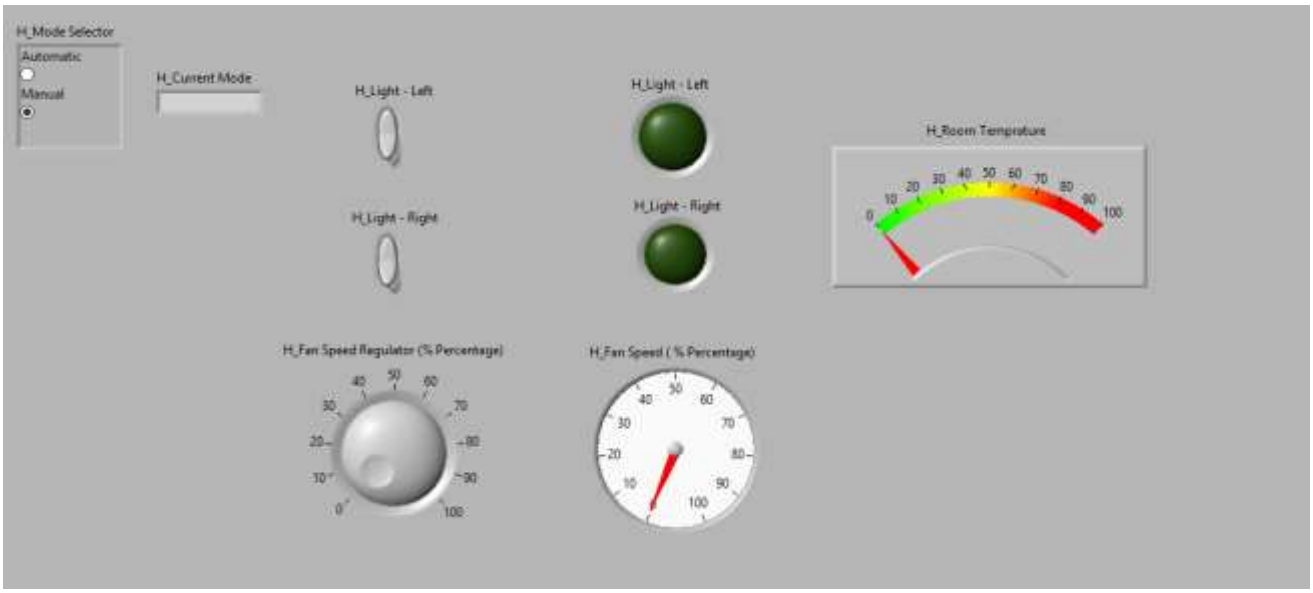
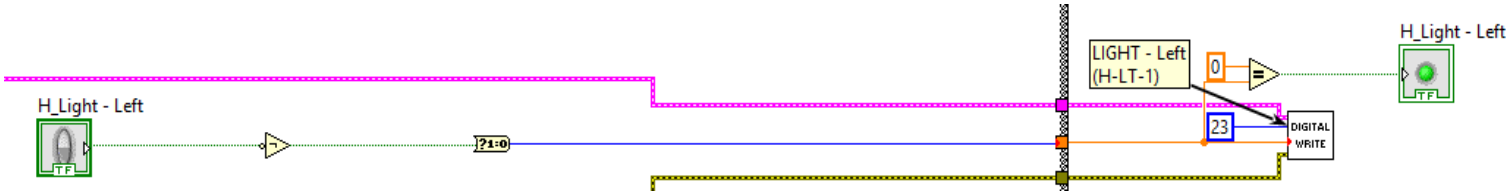
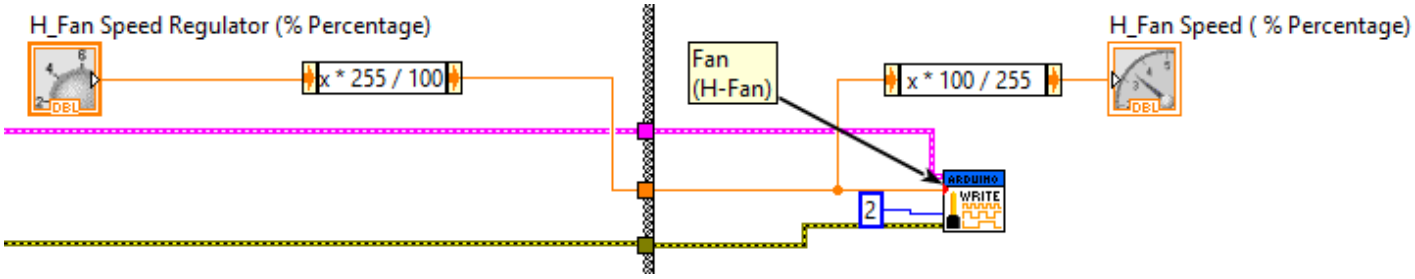


FIG 5.10 Hall front panel



When user presses hall light switch H_Light-left in above front panel, the not gate converts it into logic 0 which is digital written at pin 23(relay pin of light) and thus light turns on which is shown as indicator in front panel.



When user turns on knob of H_Fan Speed Regulator, fan starts to rotate **irrespective** of temperature reading shown by LM 35. The speed of fan can be controlled by regulator.

Fan speed is from 0 to 100 and so it has to converted into frequency of PWM pulse.

Conversion:

(x*255) / 100 = Output

Where;

x is % input from fan regulator

Output is PWM frequency

This PWM frequency is digitally written over PWM pin of Arduino by **Digital write SUB VI.**

Automatic

In automatic mode none of the switches will work and entire operation will be performed automatically.

Light will be turned ON/OFF depending on presence of person AND day/night.

Speed of Fan will vary according to temperature reading given by LM 35.

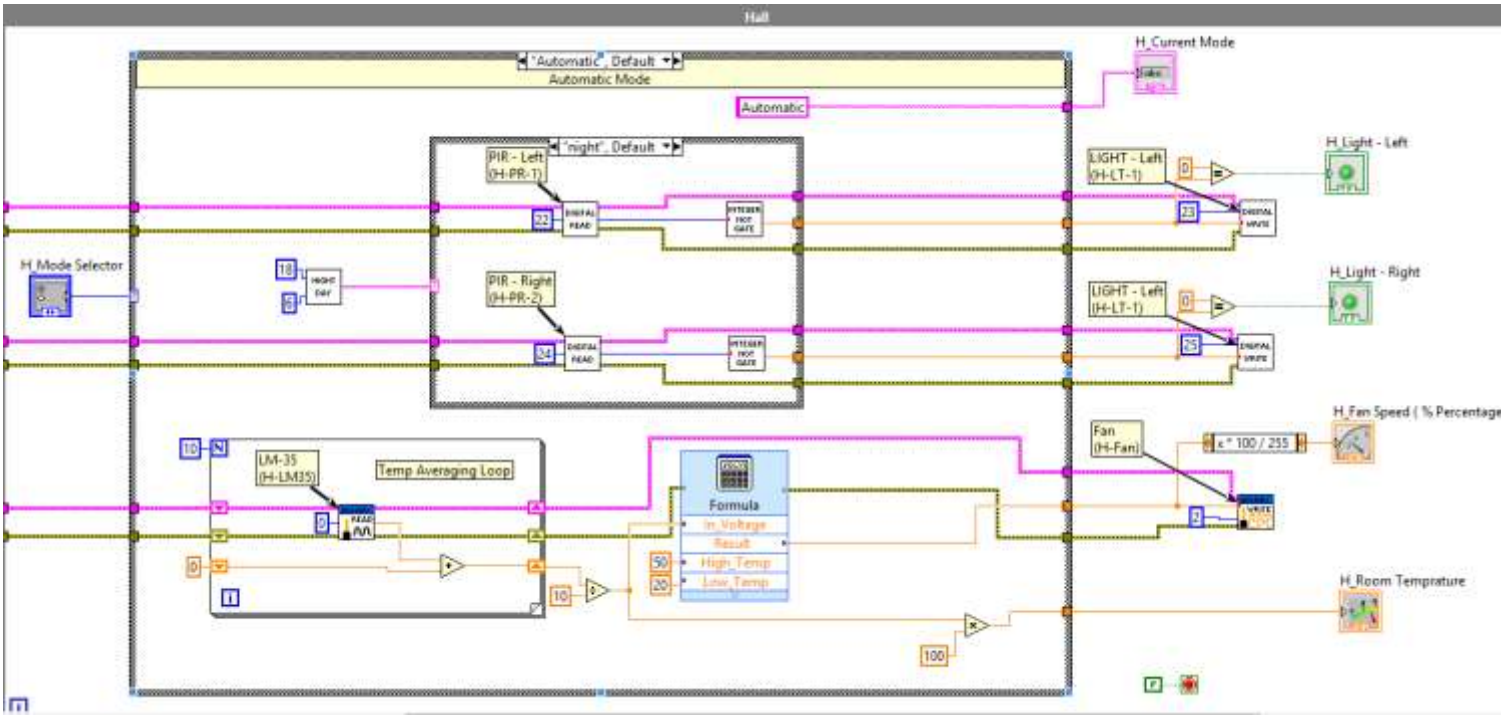
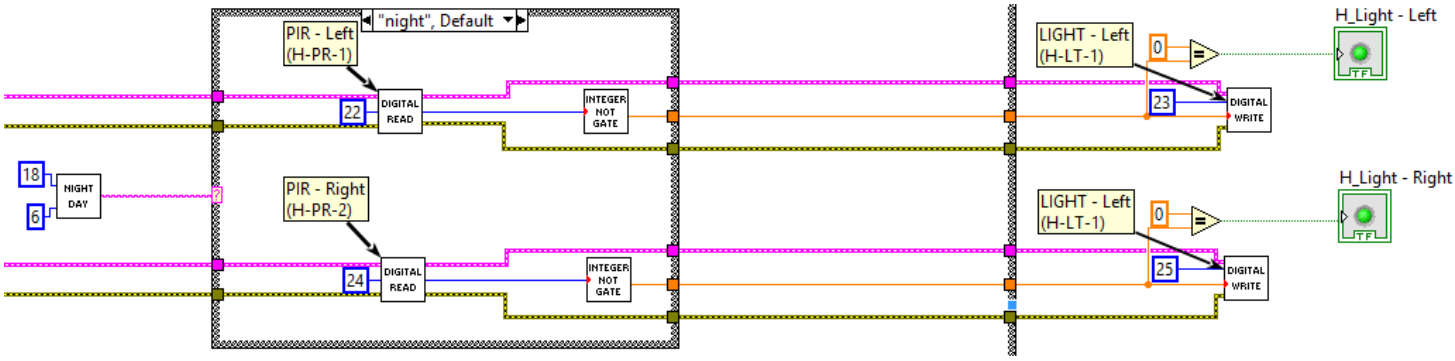


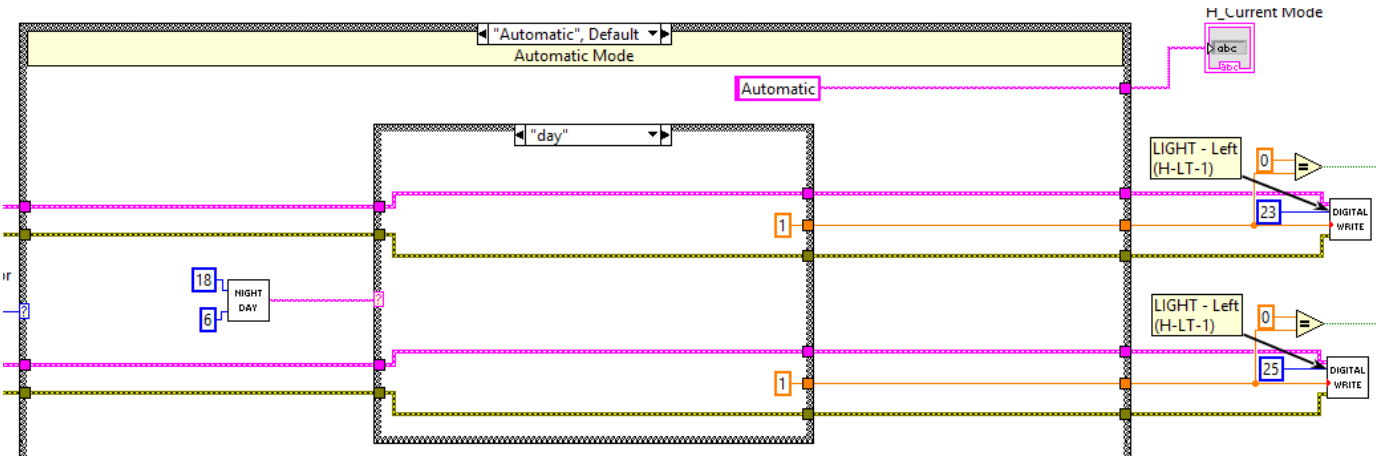
FIG 5.11 Hall automatic block diagram (See Appendix Fig-F)

Client will see same front panel as in manual mode though neither switches nor regulator will work.



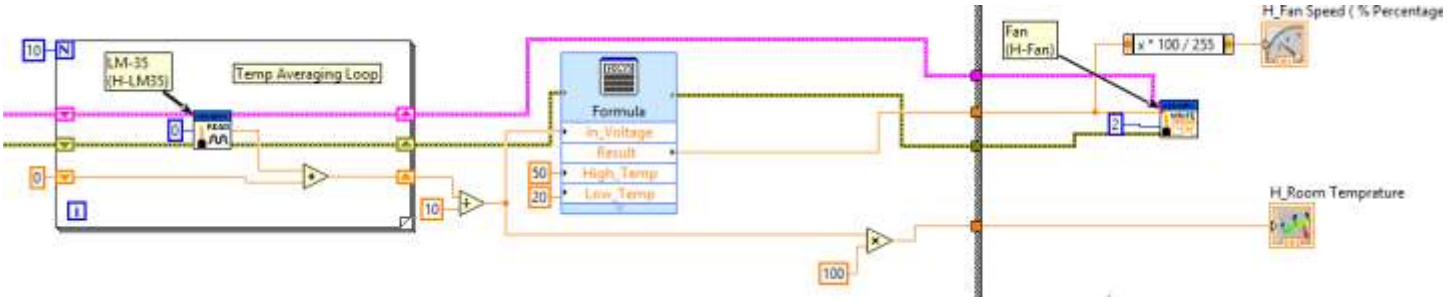
Lights will turn on only in NIGHT AND in presence of a person in hall. DAY/NIGHT sub VI will decide day or night mode of operation of above case structure. In night mode on presence of a person will be detected by PIR and Digital read sub VI will read its output and digital write sub VI will write **"0"** to relay input and thus light of that PIR will turn on.

PIR can be calibrated for sensitivity and time period of its output.



Above is DAY case of VI. During day PIR output won't matter and lights won't turn on since Digital write sub VI will write **"1"** to relay input.

Below is the hall fan automatic mode. Here speed of fan will be determined by temperature reading taken by LM 35. For better results readings of LM 35 are averaged upto 10 measurements and that averaged value is used to vary fan speed. **Analog read** is used to read LM 35 and **Temperature averaging loop** is used to carry out average.



Temperature Range is kept **20 ° C- 50 °C.**

Following expression is used to convert average input voltage of LM 35 into desired **PWM output** for fan.

$$\text{PWM OUTPUT} = \frac{255^{*}\,(\text{In\_Voltage}*100-\text{Low\_Temp})}{(\text{High\_Temp}-\text{Low\_Temp})}$$

Resulting PWM output between 0-255 is written by analog write on PWM pin of Arduino.

Front panel will have to indicators %fan speed and Room temperature in deg C. Percentage fan speed can be obtained by following expression:

%fan speed = **(x*100)/255** ;

Where;

x is PWM output

And room temperature can be obtained from readings of LM 35 as follows:

Room temperature = **x*100** ; where x is averaged voltage reading of LM35

5.2.3 KITCHEN VI

Kitchen VI makes use of **Day/Night, digital read, digital write, path, EMAIL, date and time, integer not gate** sub VI's. Apart from PIR it consists of **MQ 135** air quality sensor. Details of MQ135 are given in hardware section and we have used its digital output to ring the buzzer.

VI of kitchen has two modes of operation which are:

- Manual
- Automatic

TABLE 5.2.3 Kitchen input output

Sr No.	Digital input	Digital output
1	PIR	Light
2	MQ135	Buzzer, Mail

Manual

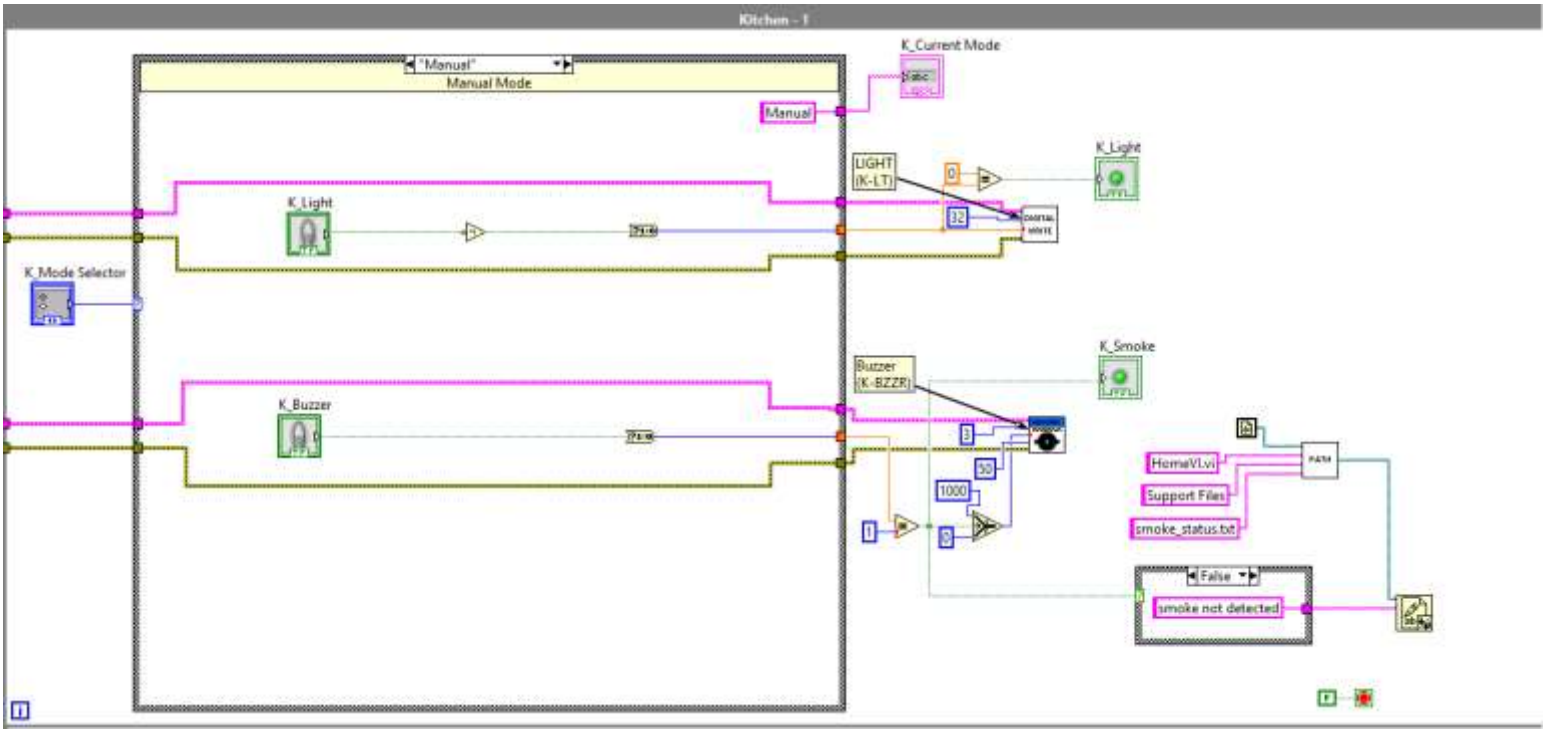


FIG 5.12 Kitchen manual block diagram (See Appendix Fig-G)

Client won't see above window nor will he be able to change the VI. Client will see following front panel on his screen.

Working of PIR and light is same in all VI's in both manual and automatic modes. Automatic mode used DAY/NIGHT sub VI **and** PIR output to switch on the light while manual mode uses only switches to turn on light irrespective DAY or NIGHT.

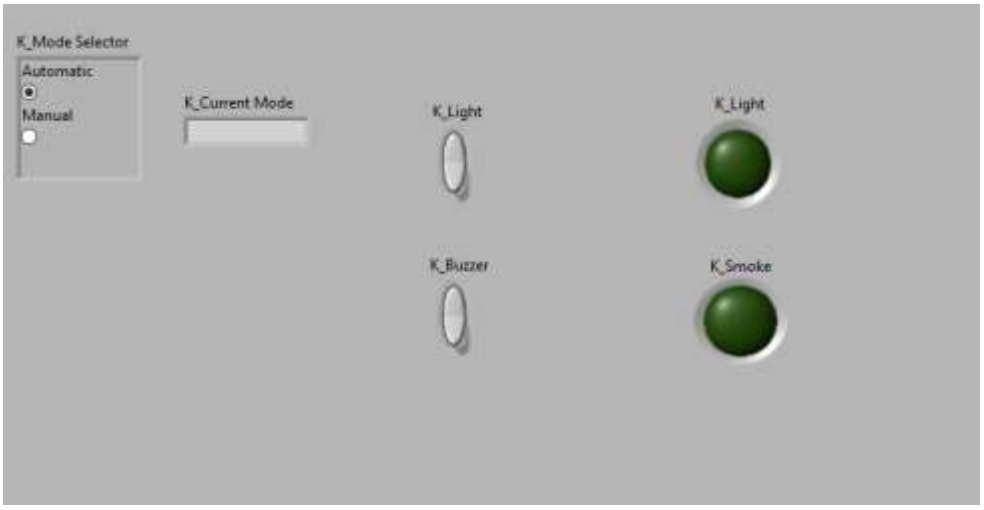


FIG 5.13 Kitchen front panel

Automatic

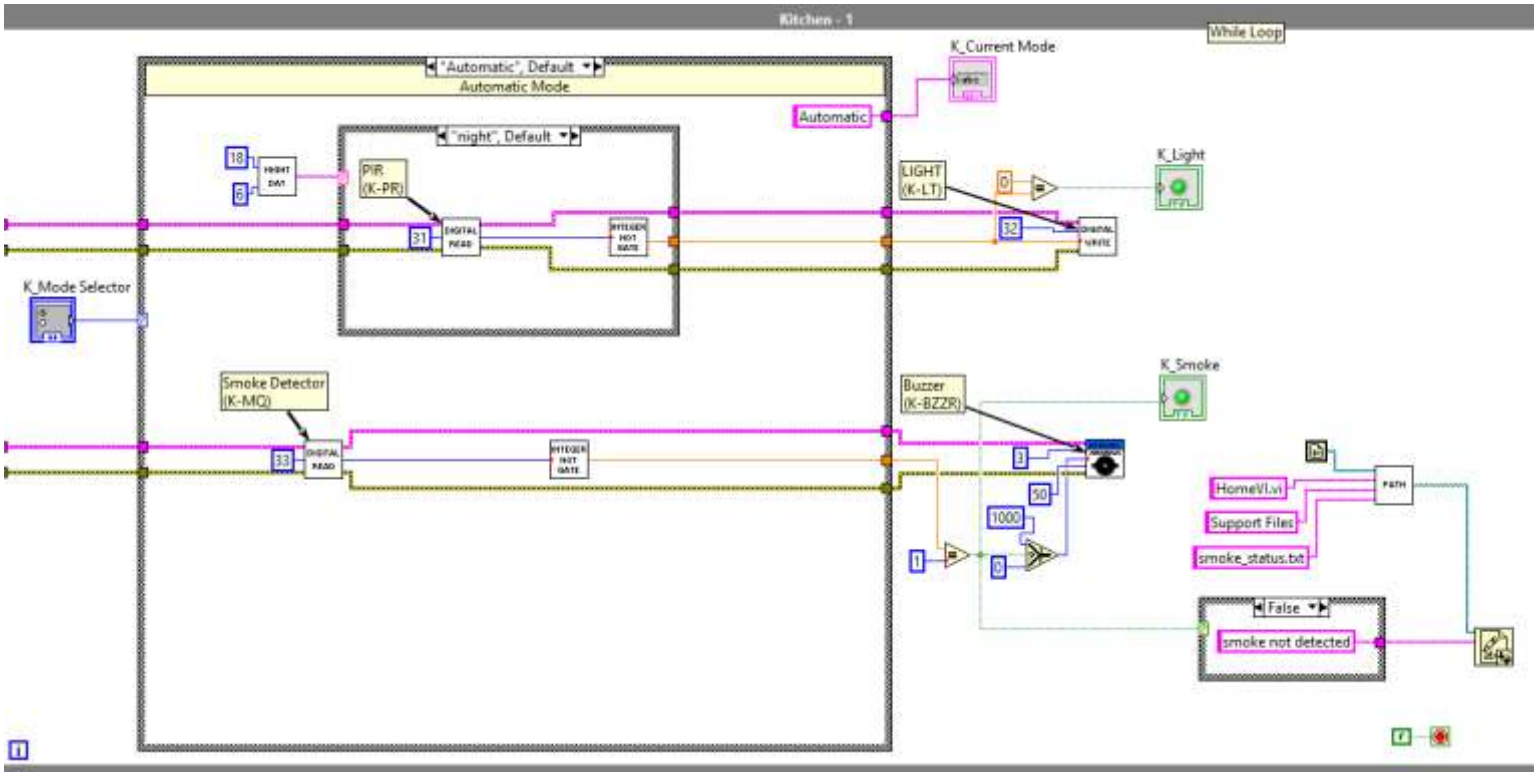
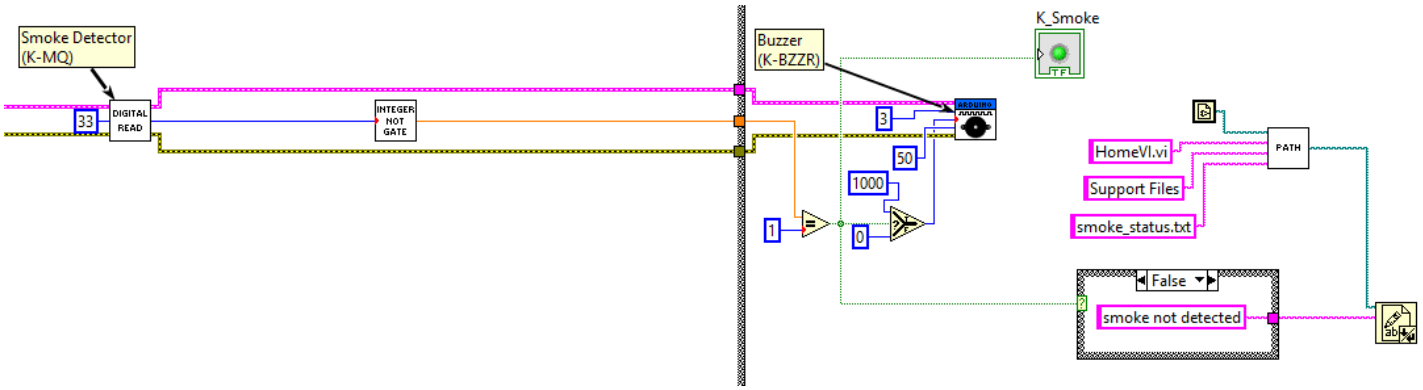
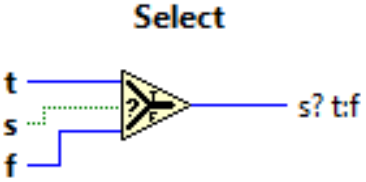


FIG 5.14 Kitchen automatic block diagram 1 (See Appendix Fig-H)

Smoke detection is carried out using MQ135. VI of smoke detection is explained below.



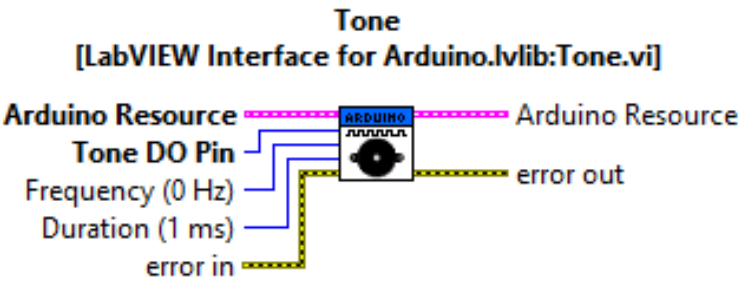
MQ135 gives 0V output upon detection of smoke. Integer not gate sub VI converts 0 to 1 and so if condition is met select block will give 1000Hz as frequency to tone generator block or else it will give 0Hz on false condition.

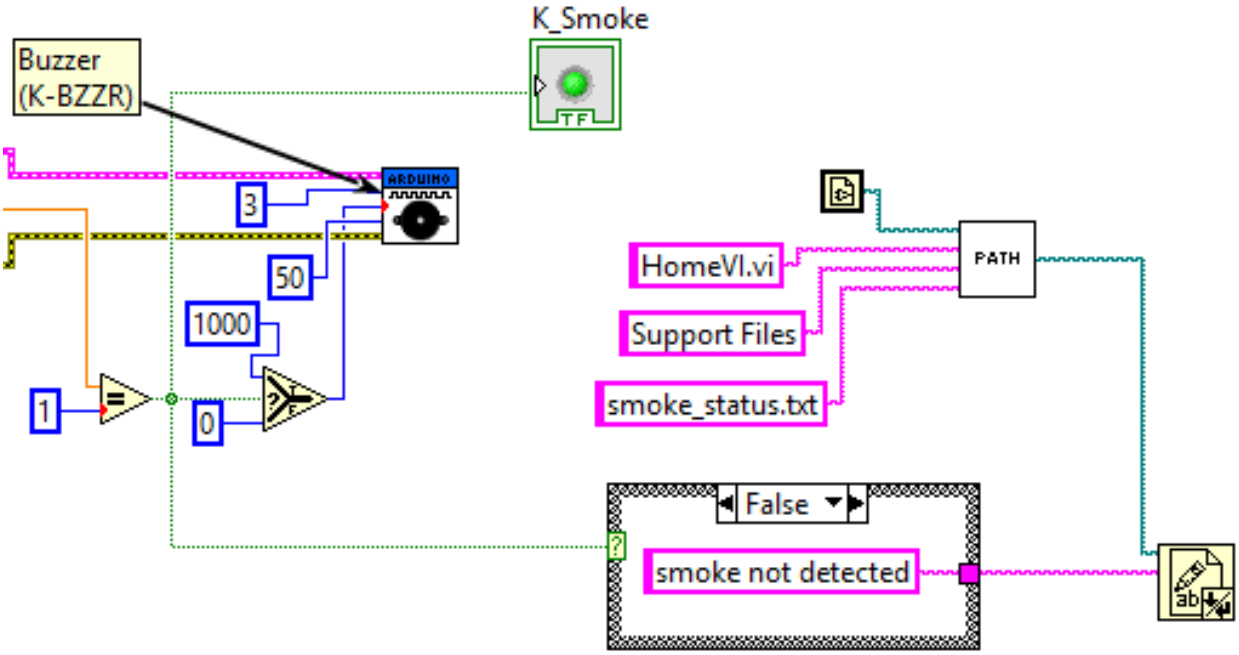


Tone generator block is used to give PWM signal to buzzer to detect smoke.

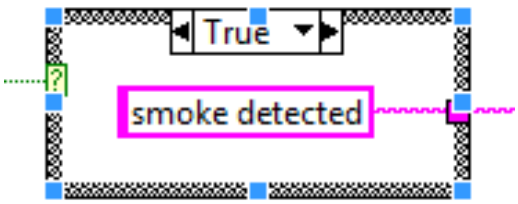
Tone DO Pin is PWM pin of Arduino connected to buzzer.

Frequency is given by select block upon detection of smoke. (either 1000Hz or 0Hz)





Above VI is used to create a text file at specified path in **support files** folder. The content of the text file will depend on output of MQ135. If MQ135 output is 1 that is no smoke then “smoke not detected ” will be written in smoke_status.txt file.



However if output is 0V then “smoke is detected ” will be written in smoke_status.txt file. Now this text file will be read and if the strings compared matches then mail will be sent to owner specifying DATE and TIME of smoke detection. This will be done by following VI.

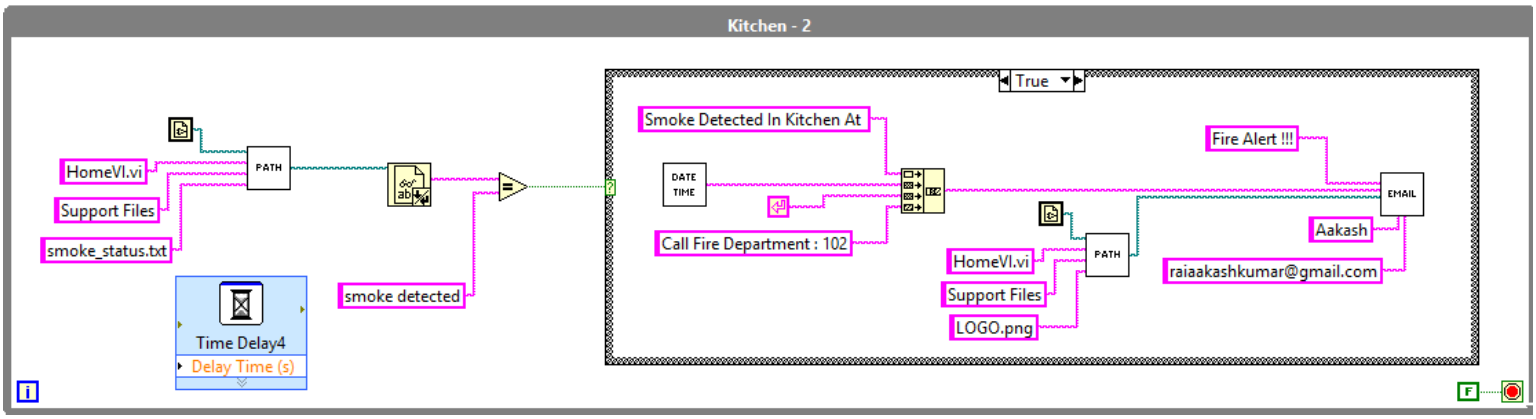
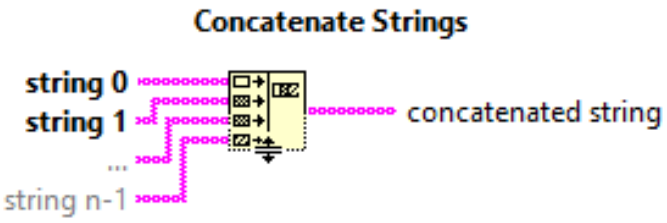
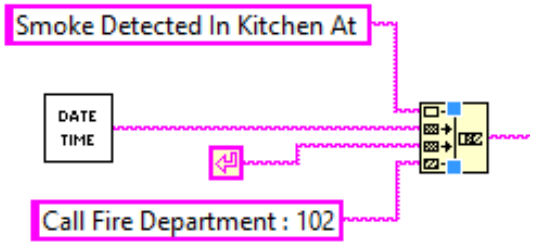


FIG 5.15 Kitchen automatic block diagram 2 (See Appendix Fig-I)

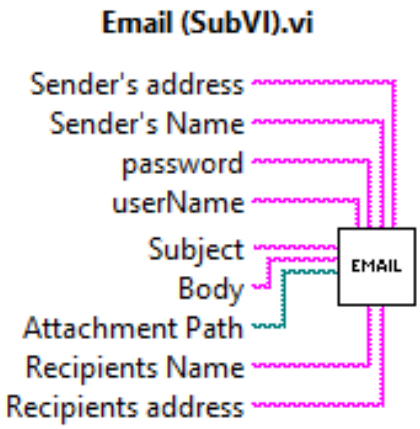
If the string in smoke_status.txt file matches "smoke detected" then case structure true case will be executed.





Concatenate string block will concatenate strings "Smoke Detected in Kitchen At" "Date and time". Then Email sub VI will send this concatenated string in mail of owner of the house.

Email sub VI has to be given subject name of mail, recipients name, recipients address.



5.2.4 BEDROOM VI

Bedroom VI uses Digital read, Digital write, Day and night and Integer not gate sub VI's. PIR, light and room temperature are main functions of bedroom VI. Like others it has two modes of operations namely:

- Automatic
- Manual

TABLE 5.2.4 Bedroom input output

Sr No.	Digital input	Digital output
1	PIR	Light

Sr No.	Analog input	Analog output
1	LM 35	Temp meter

Manual

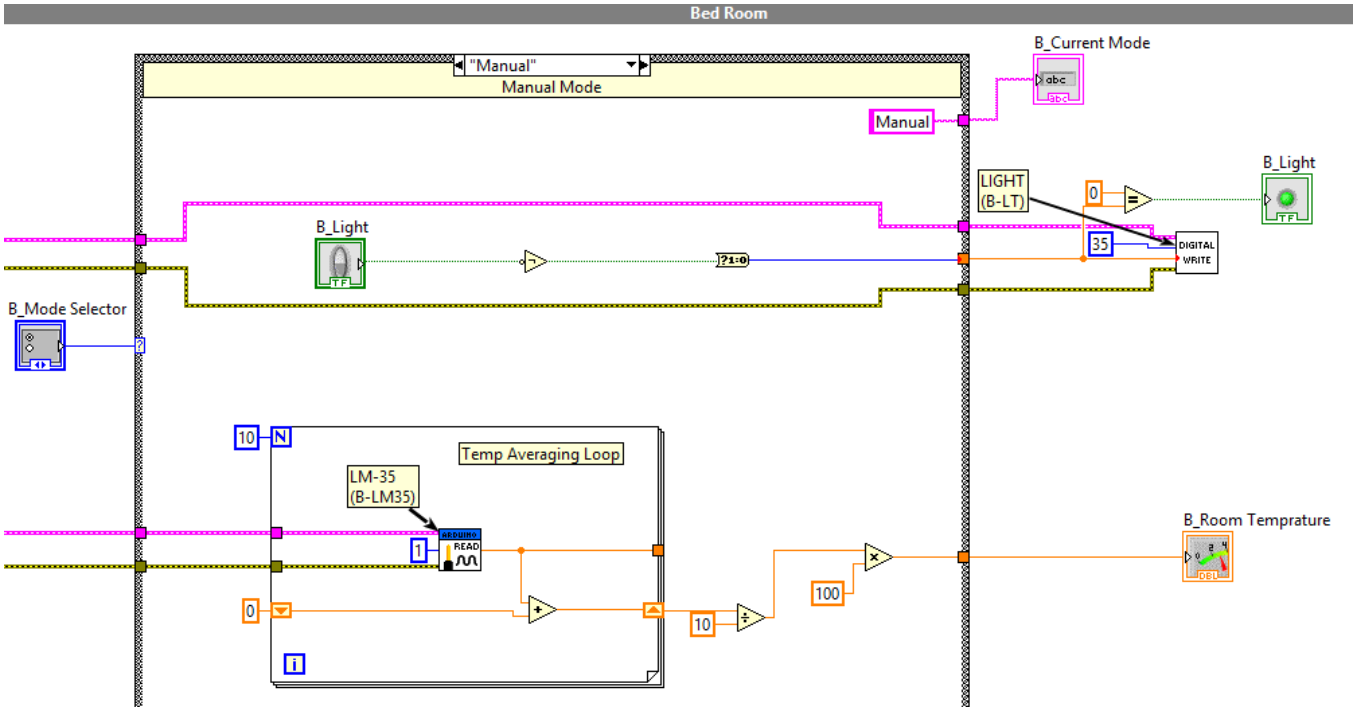


FIG 5.16 Bedroom manual block diagram (See Appendix Fig- J)

Working of PIR and light in manual mode is same as hall. Here below front panel shows what client will see and temperature indicator shows temperature of bedroom.

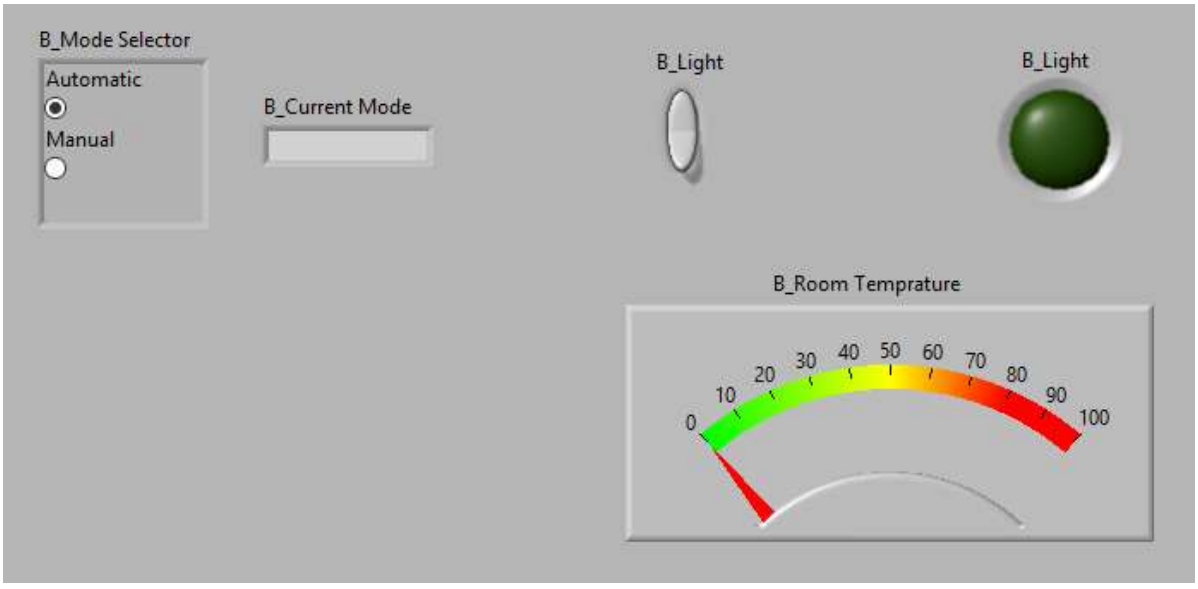


FIG 5.17 Bedroom front panel

Automatic

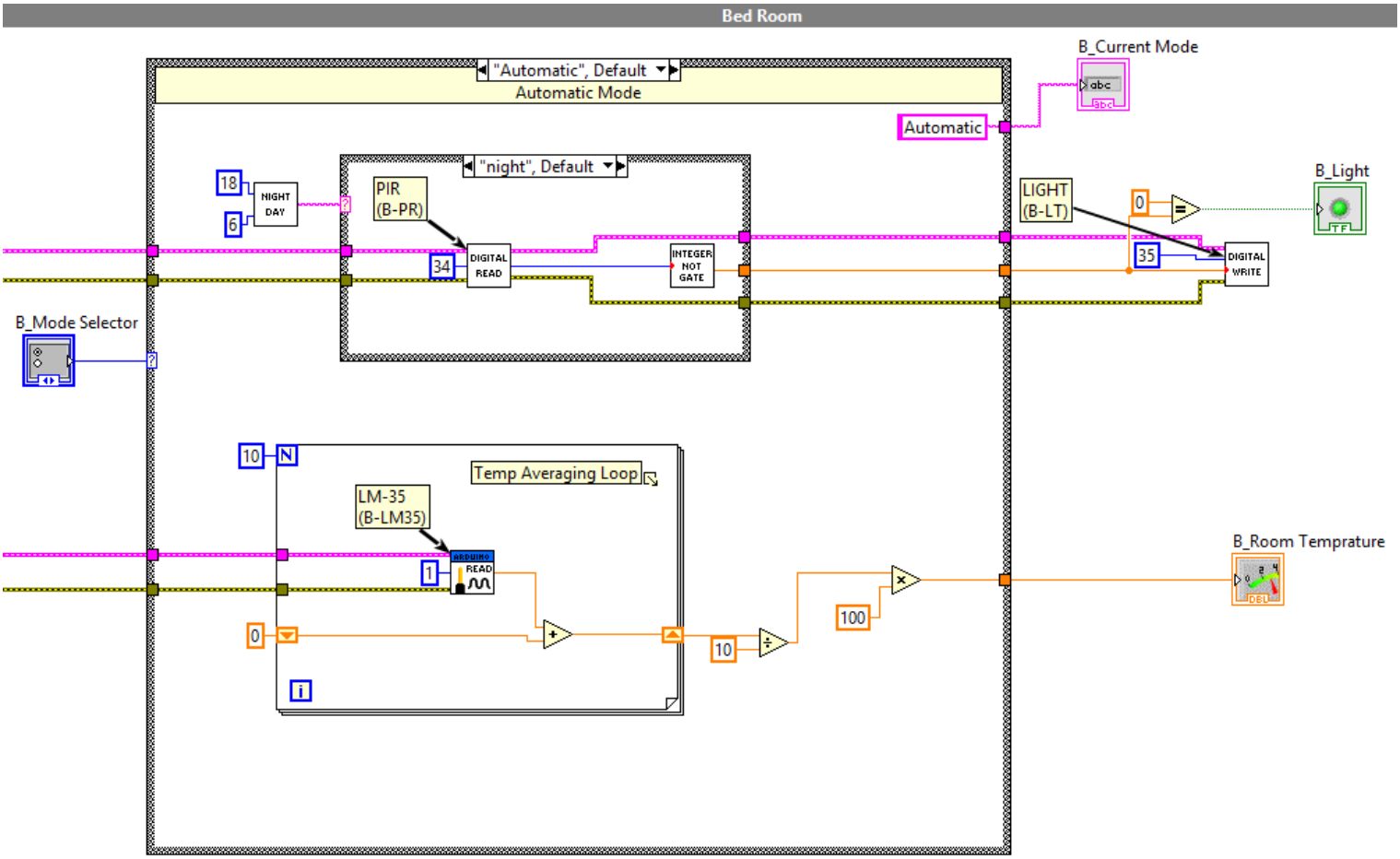


FIG 5.18 Bedroom automatic block diagram (See Appendix Fig-K)

5.2.5 WASHROOM VI

Washroom VI uses **Digital read, digital write, integer not gate, day and night and path** sub VI's**.** VI operates in two modes:

- Automatic
- Manual

TABLE 5.2.5 Washroom input output

Sr No.	Digital input	Digital output
1	PIR	Light

Sr No.	Analog input	Analog output
1	LM 35	Temp meter

Manual

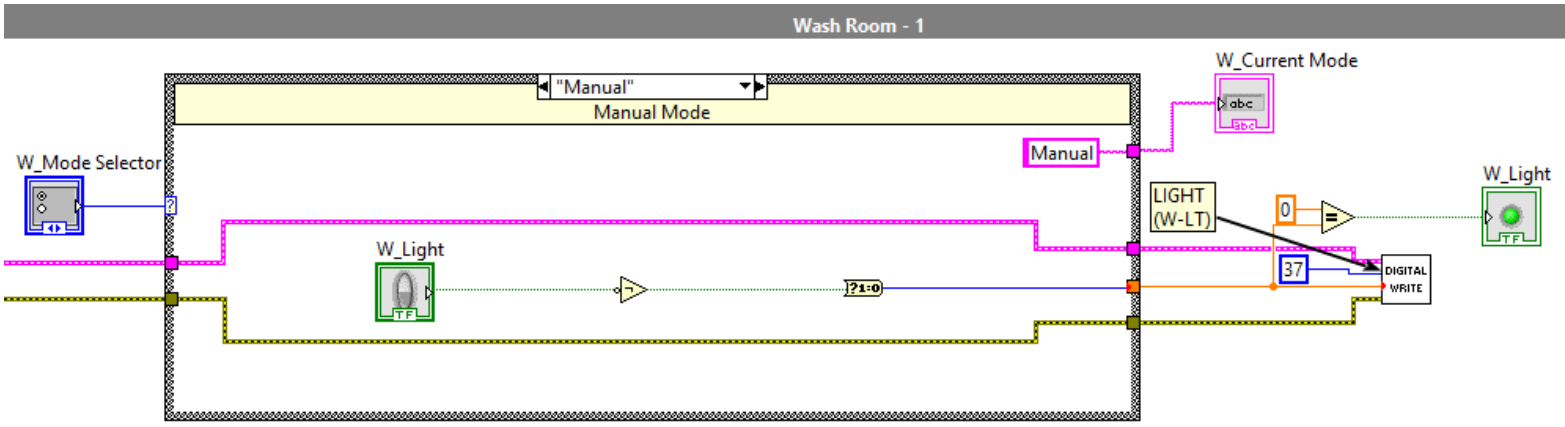


FIG 5.19 Washroom manual block diagram 1 (See Appendix Fig-L)

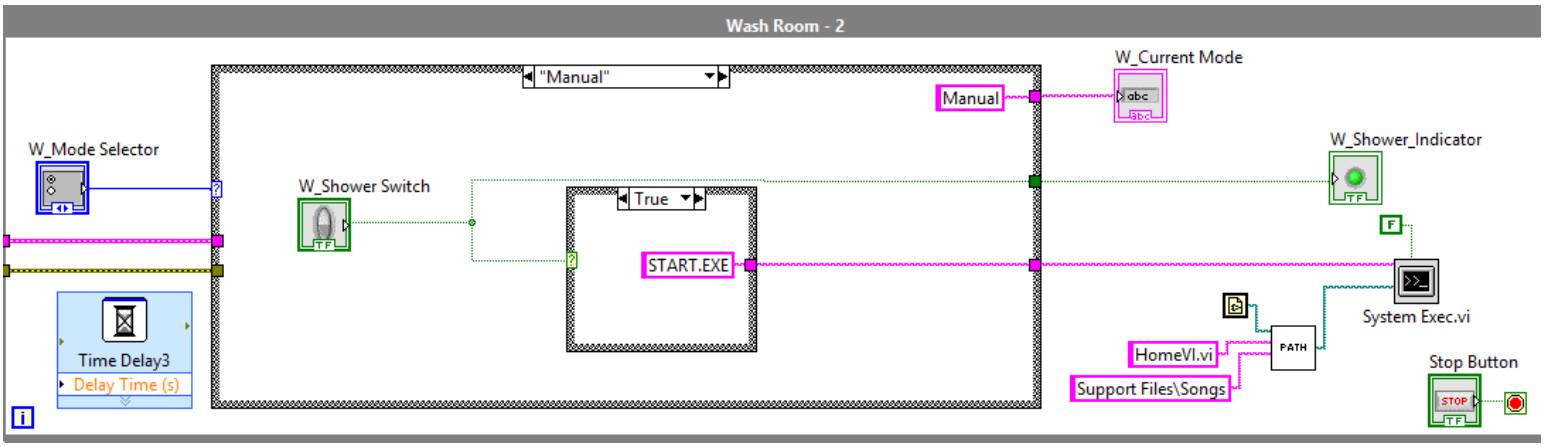
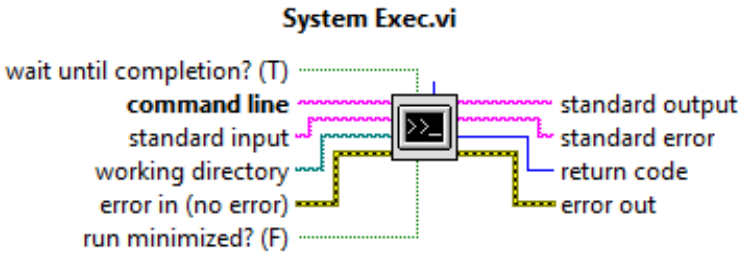


FIG 5.20 Washroom manual block diagram 2 (See Appendix Fig-M)

Fig 5.2.5(a) is VI of PIR and light in manual mode while FIG 5.2.5(b) is VI of music in manual mode.

We like to hear music while bathing so we have provided this feature in our home. In manual mode when music switch is pressed music turns on. We have used **SPDT toggle** switch as music switch.

The system.Exec block will execute command START.exe, which has batch files, when music switch is turned on. Now system.Exec block need address of the START.exe file and that address is provided using PATH sub VI.



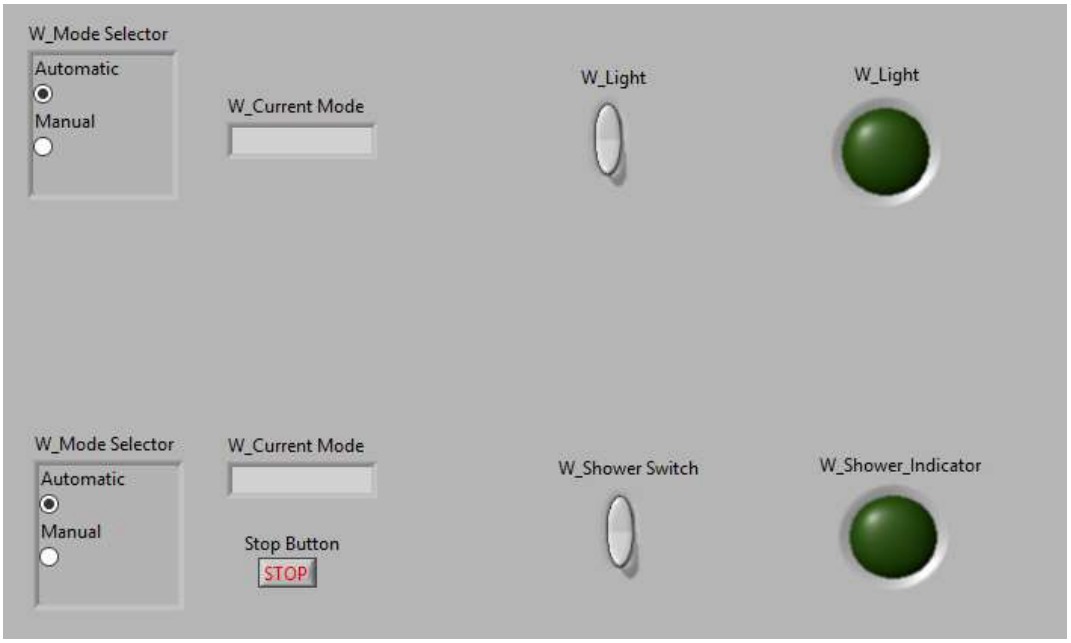


FIG 5.21 Washroom front panel

Front panel shows light and shower switch indicator.

Automatic

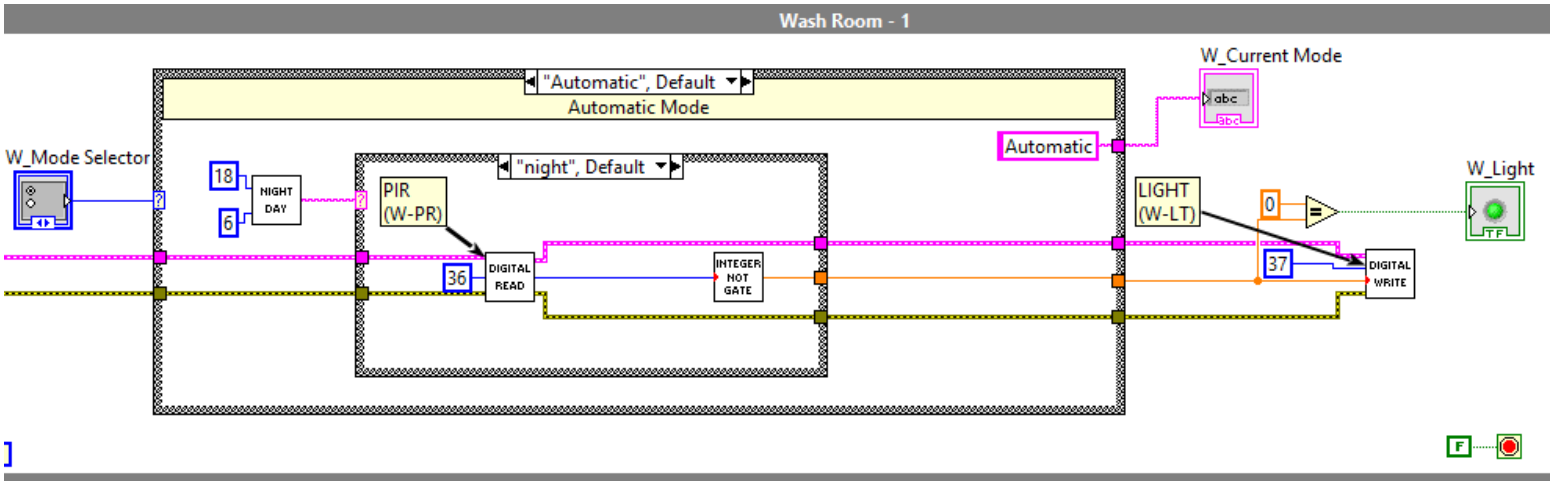


FIG 5.22 Washroom automatic block diagram 1 (See Appendix Fig-N)

Working of PIR and light is same and below is VI to play music in automatic mode. For fully automatic mode we need to use **flow switch** but due to practical reasons we have not used flow switch. Instead we have used a simple toggle switch to indicate flow of water through shower and named it as shower switch.

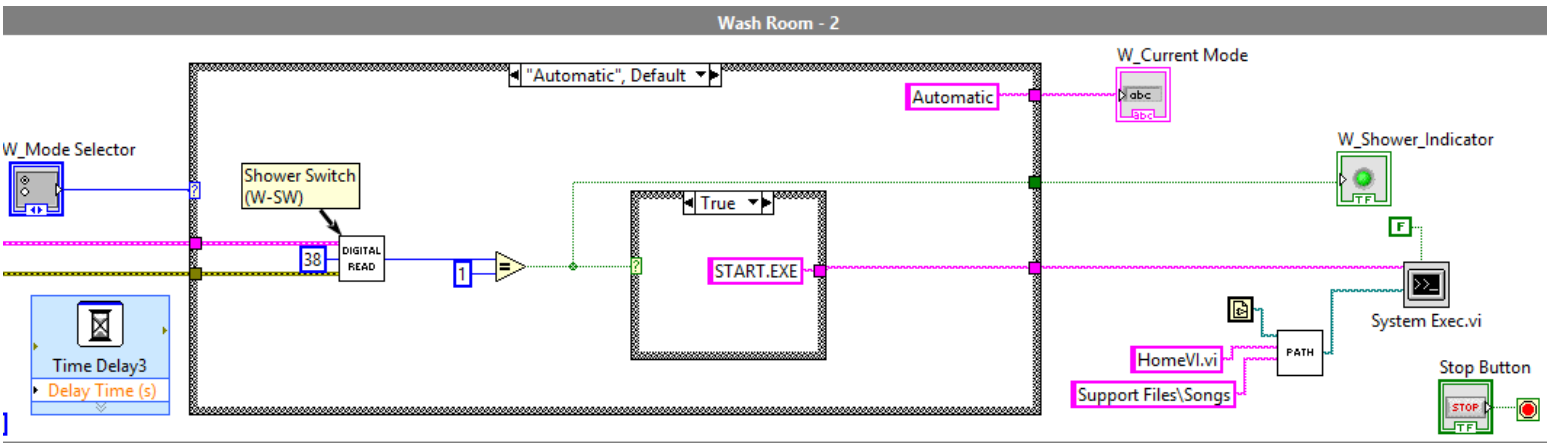


FIG 5.23 Washroom automatic block diagram 2 (See Appendix Fig-O)

5.2.6 MAIN DOOR SECURITY

Main door is password protected. So if the person writes right password only then the door will open. We have provided 3 attempts to enter correct password. When third attempt is wrong too, person's image will be captured and a mail will be sent to the owner of the house with a subject **Intruder alert.** Mail will contain attachment of intruder's photo captured by CCTV and date and time of intrusion.

Password VI has 3 cases in case structure. This are following:

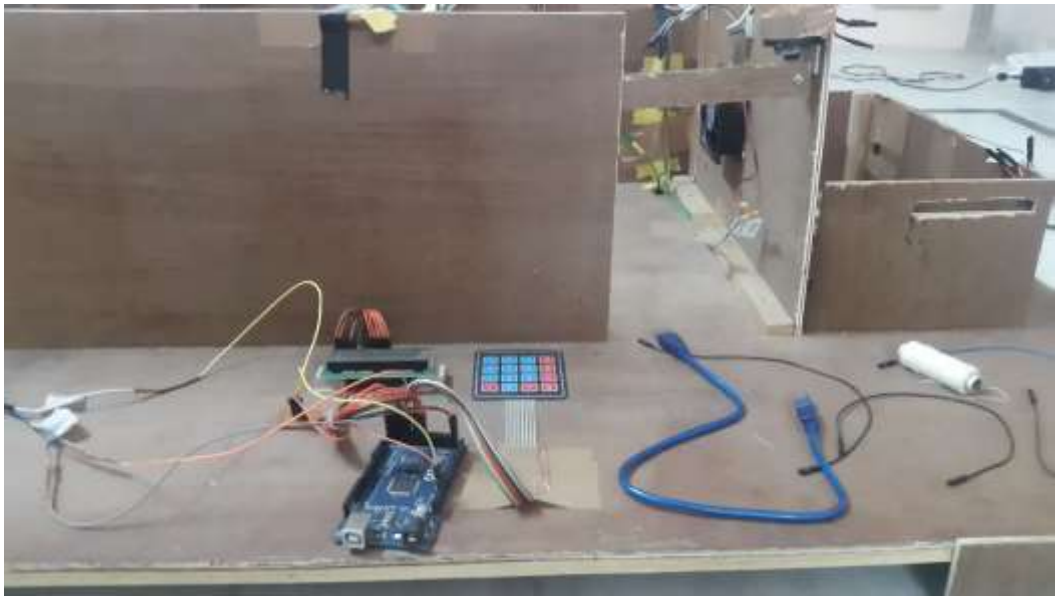
- Normal condition
- Right password
- Wrong password

Slave Arduino program checks the given input to the keyboard and gives output to the 46 and 47th pins of master Arduino. The state if the 46th and 47th pins depends upon password entered.

TABLE 5.2.6 Input output table of main door security

Sr No.	Password Status	46th pin	47th pin
1	Correct	1	1
2	Wrong(0 attempts left)	0	0
3	Normal condition	0 or 1	1 or 0

All above 3 states has different three VI's and each of these VI's has different function. Correct password VI has function of opening and closing the front door. Wrong password VI has sends the image of the intruder in attachment to the owner of the house through mail.



Normal condition

FIG 5.24 Main door security normal condition (See Appendix Fig-P)

Normal condition indicates either of following things:

Password has not been entered **OR** Attempts to write correct password are left.

In both cases we need not to do anything and so normal condition VI has "**do nothing"** in it.

Right password

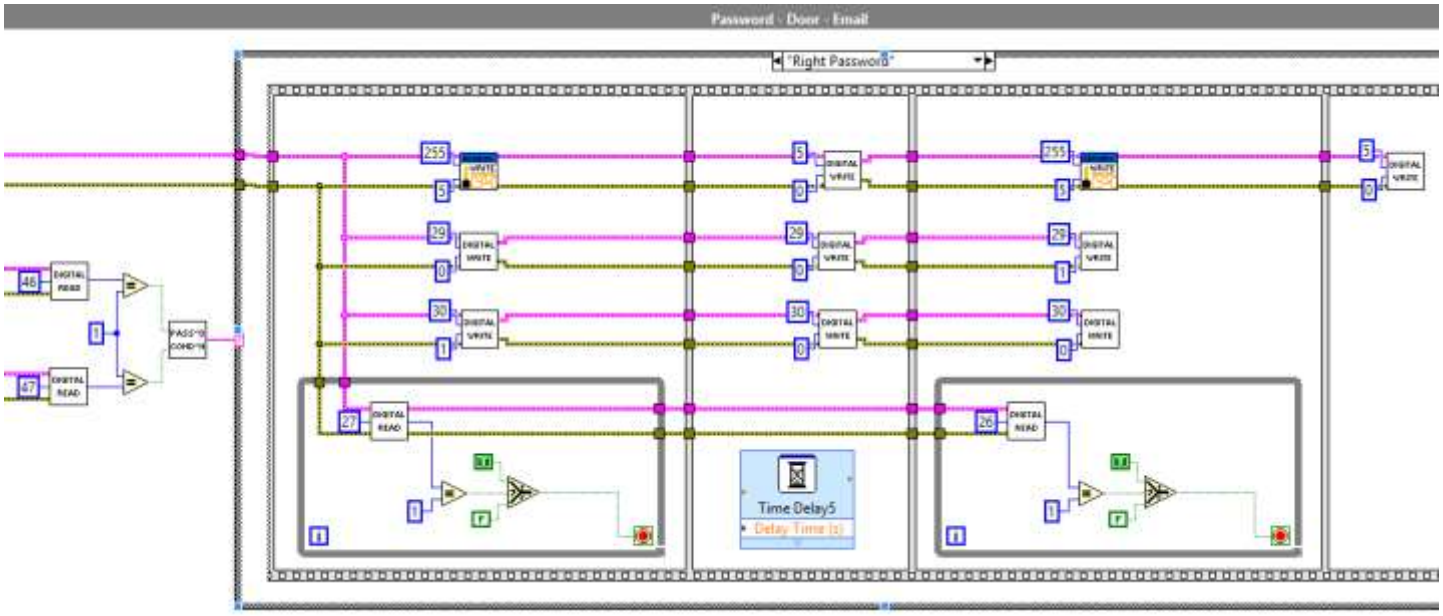


FIG 5.25 Main door security right password condition (See Appendix Fig-Q)

Right password case of "Password" VI functions when correct password is entered through keyboard. If the password entered is correct then front door opens. Functioning of front door is controlled by motor which indeed is controlled by motor driver. Limit switches are used to control the operation of motor.

When entered password is correct then door will be opened by motor until back limit switch is pressed by the door. As soon as limit is pressed by door motor stops and so door remains open. We have kept a delay of 20 sec between opening and closing of door. So door will remain open until 20 sec after which motor will run un reverse direction and door will get closed. Motor will run until front limit switch is pressed by closing door. Once door closes it wont open unless correct password is entered.

Digital read, password condition and digital write are sub VI's used in right password case of password VI.

Wrong password

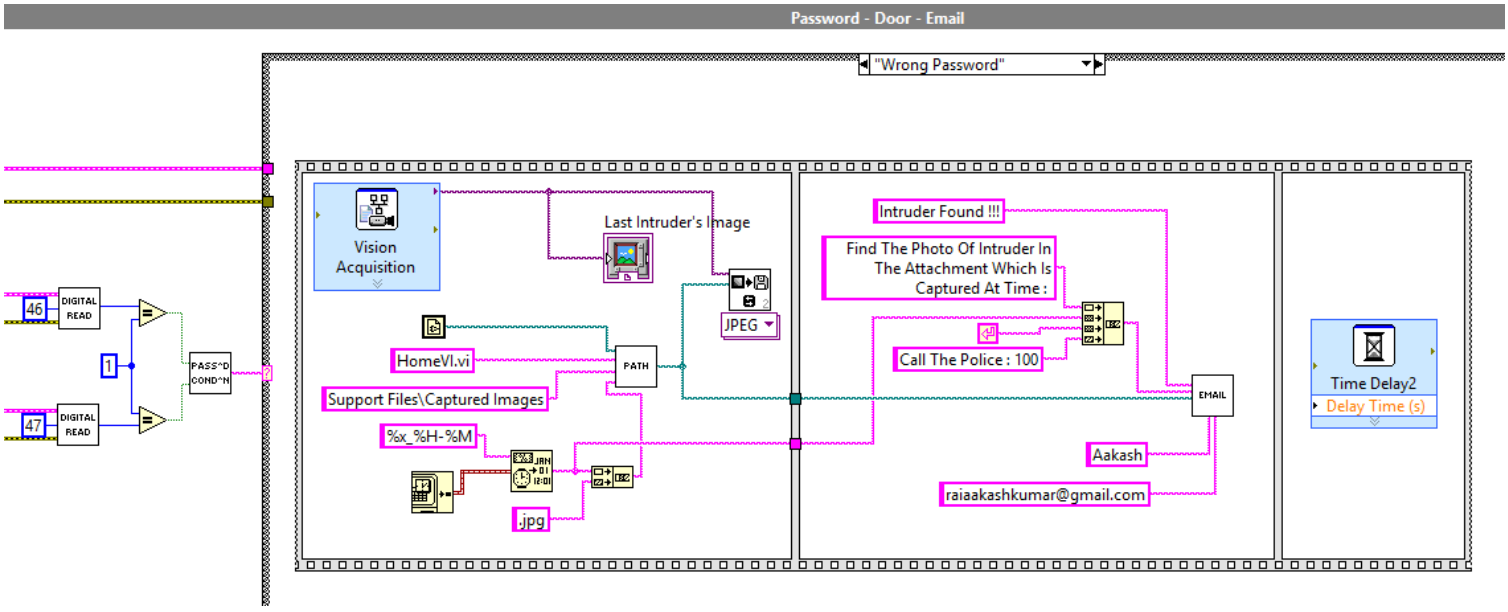
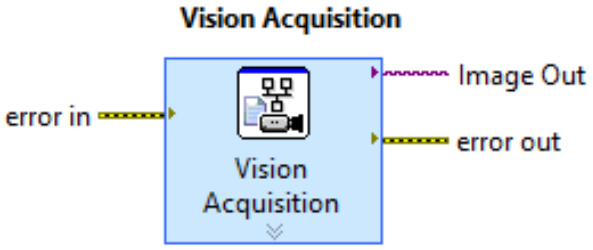


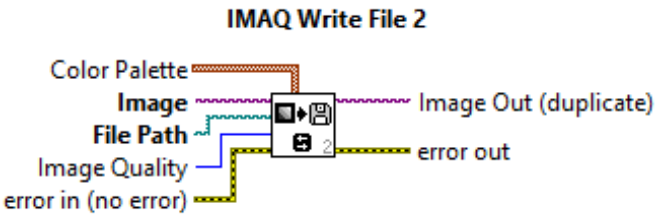
FIG 5.26 Main door security wrong password condition (See Appendix Fig-R)

Wrong password condition of password VI is particularly kept for security purpose. We have provided **3 attempts** to enter correct password. When 3rd attempt goes wrong above VI operates.

Vision library is needed for above VI to work.



Vision acquisition block will acquire image through web cam or attached camera.



Writes an image to a file in JPEG format. It requires file path which is given by **path** sub VI and creates a duplicate image in required format.

Path sub VI provides path of image captured by vision acquisition and IMAQ converts that image in JPEG format. Get date/time in seconds provides current date and time. This along with path will form attachment path and is given to email sub VI. Email subject will be intruder alert.

Email can be sent to one person only and so owner of the house will receive all alert mails. Though recipient can be changed. Also image of intruder will appear in following front panel.

FIG 5.27 Intruder's image(Main door security front panel)

5.2.7 MAIL BOX

Whenever mail arrives, letter trips the limit switch kept in mail box. VI will then send email to the owner of the house.

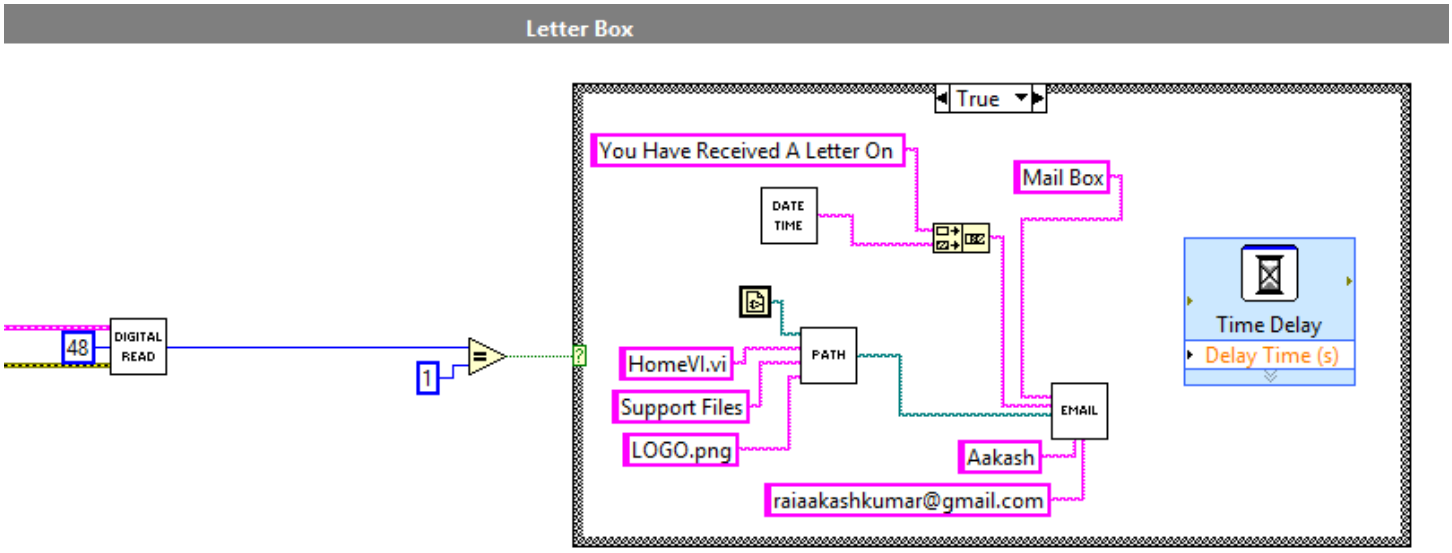


FIG 5.28 Mail box block diagram (See Appendix Fig-S)

Limit switch output is digitally read and when limit switch trips true case will be executed. False case has do nothing in it. Mail body would have words “you have received a letter on DATE AND TIME”. Again the recipient can be changed.

5.3 ARDUINO PROGRAMMING

Most of the programming is done in LabView only. Arduino programming is used to only to install LIFA and keypad read. We have used two Arduino’s one is primary other is secondary. Secondary Arduino programming is done in Arduino. Keypad read and LCD

program is installed in it. This was done because single Arduino was not having enough pins and including a separate Arduino makes job easily done.

Firmware

Below is the screen shot of firmware used for LabView Arduino interfacing. This has to be installed in primary Arduino only.

FIG 5.29 LIFA base

This firmware is too long and only one screenshot is shown here.

6 CONCLUSION AND FUTURE SCOPE

Web based smart home is home automation which can be controlled via web by its user. Its application has no limits since we can always add new features in it. Web control provides easy way for its users to control and monitor their house. It can be concluded that entire concept doesn't cost too much and is reliable. Also availability of software and hardware used is easy. User can himself add new features inspite of limited knowledge. Graphical programming makes it easier to understand and change program whenever needed. Minimal use of codes makes it simpler.

Use of PLC as controller and SCADA as HMI was easily possible but it is much more costly compared to Arduino + LabView. Though LabView provide excellent programming options it has week user interface when compared to SCADA. PLC + SCADA provides better control and user interface at the cost of money.

While making this project we have learned many new concepts and enhanced our knowledge in LabView. We have tried everything possible in our effort to make this system best. Such concept is still in its maturing stage and a lot can be expected in near future.

FUTURE SCOPE

With limited components and flexibility we have made this entire project with optimized features in it. However this can be bettered and creative things can be added.

We can always add new features according to the requirement of the user. There are no limitations of adding new features in house. Following are some of the features that can be added:

- Air condition control
- Curtain control
- Energy meter
- Solar applications

Apart from features, we add concepts like “INTERNET OF THINGS” in which all devices and components are connected to web and one thing controls other without human intervention. Though it might get costly now, but it is possible in near future.

REFERENCES

- www.ni.com
  www.instructables.com
- www.wikipedia.org
- www.ti.com/lit/ds/symlink/lm35
- https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135
- www.ladyada.net/media/sensors/PIRSensor-V1.2

Books

- LabVIEW for data acquisition by Bruce Mihura
- LabVIEW introduction course manual by NI
- Programming Arduino with LabVIEW by Marco Schwartz and Oliver Manickum