**PHENIKAA UNIVERSITY**

**FALCULTY OF ELECTRONIC ENGINEERING**

Logo, company name

Description automatically generated

**BASIC PROGRAMMING FOR ELECTRONICS FINAL-TERM PROJECT REPORT**

**Project: Smart System utilize passive Infrared sensor to Control Fan and Light in smart home**

1. Duong Doan Tung

2. Nguyen Trong Huy Hoang

3. Le Hoang Nam

**HA NOI, 12/2022**

TABLE OF CONTENTS

[WORKLOAD DISTRIBUTION 4](#_Toc121033755)

[Section 1: ABSTRACT 1](#_Toc121033756)

[Section 2: PROJECT OVERVIEW 2](#_Toc121033757)

[1. Inspiration 2](#_Toc121033758)

[2. Basic Idea 2](#_Toc121033759)

[Section 3: PROJECT DETAILS 3](#_Toc121033760)

[1. Hardware 3](#_Toc121033761)

[1.1. Electronic parts 3](#_Toc121033762)

[1.1.1. USB-6255 3](#_Toc121033763)

[1.1.2. HC-SR501 3](#_Toc121033764)

[1.1.3. L298N 3](#_Toc121033765)

[1.1.4. Light Emitting Diode 3](#_Toc121033766)

[1.1.5. DC Motor 3](#_Toc121033767)

[1.2. Wiring diagram 4](#_Toc121033768)

[1.3. House model design 4](#_Toc121033769)

[2. Software 4](#_Toc121033770)

[2.1. Labview 4](#_Toc121033771)

[2.1.1. User Interface 4](#_Toc121033772)

[2.1.1.1. Manual control 5](#_Toc121033773)

[2.1.1.2. Live feed tab 5](#_Toc121033774)

[2.1.1.3. History tab 6](#_Toc121033775)

[2.1.1.4. Schedule tab 7](#_Toc121033776)

[2.1.1.5. TCP tab 8](#_Toc121033777)

[2.1.1.6. ORM tab 8](#_Toc121033778)

[2.1.1.7. Configuration tab 9](#_Toc121033779)

[2.1.2. Labview block diagram 9](#_Toc121033780)

[2.1.2.1. Program structure 9](#_Toc121033781)

[2.1.2.2. Main VI block diagram 10](#_Toc121033782)

[2.1.3. SubVIs 18](#_Toc121033783)

[2.1.3.1. Error collector 18](#_Toc121033784)

[2.1.3.2. Error handler 18](#_Toc121033785)

[2.1.3.3. Simulation check 19](#_Toc121033786)

[2.1.3.4. Retry TCP 19](#_Toc121033787)

[2.1.3.5. Win32 decode IMG stream 19](#_Toc121033788)

[2.1.3.6. String to GUID Cluster 20](#_Toc121033789)

[2.1.3.7. Console Add 20](#_Toc121033790)

[2.1.3.8. Get color 20](#_Toc121033791)

[2.1.3.9. Change mode 21](#_Toc121033792)

[2.1.3.10. Create schedule 21](#_Toc121033793)

[2.1.3.11. Sort schedule 21](#_Toc121033794)

[2.1.3.12. 22](#_Toc121033795)

[2.1.3.13. 22](#_Toc121033796)

[2.1.3.14. 22](#_Toc121033797)

[2.1.3.15. a 22](#_Toc121033798)

[2.2. Python Application 22](#_Toc121033799)

[Section 4: REFERENCES 27](#_Toc121033800)

# WORKLOAD DISTRIBUTION

|  |  |
| --- | --- |
| Full name | Task |
| Duong Doan Tung | o Python, Labview Programming   * Edit video |
| Nguyen Trong Huy Hoang | * Setup hardware * Decorate, house design |
| Le Hoang Nam | * Write report * Build hardware |

Project include:

1. Project report
2. Presentation file and video
3. Python files
4. Labview files (18 subVIs included)

Full project material can be found here: <https://drive.google.com/drive/folders/15Gyvd-AxRdvlOK2FSbTaDAAlVZT9CAch?usp=sharing>

**Project repository:** <https://github.com/dtungpka/I.C.F.L>

*Special thanks to our teacher Dr. Huy Minh Le for his help in this project*

# Section 1: ABSTRACT

Educators and researchers worldwide are using National Instruments products to automate routine tasks, accomplish new objectives, replace outdated and expensive equipment, and demonstrate students the potential of high technology. Engineers have used virtual instrumentation for more than 25 years to bring the power of flexible software and PC technology to test, control, and design applications making accurate analog and digital measurements. To demonstrate Labview's potential and demonstrate Labview's capabilities in real-world applications, in this paper, we created a Smart System that utilizes passive Infrared sensor to Control Fan and Light in smart home. This system is capable of controlling Light Emitting Diodes (LED) and DC motors through several inputs.

# Section 2: PROJECT OVERVIEW

## Inspiration

From cities and transportation to the resources we farm, emerging Internet of Thing (IoT) technologies are set to make everyday processes more integrated and easier. As the sensors, data storage, the Internet, and analytics become faster, cheaper, better, and more integrated together, users will be able to rely on smart system more. IoT devices will have a significant impact on many aspects of our lives including how we live, drive, and farm animals and crops.

The Internet of Things (IOT) technology establishes a connection between all things and the Internet via sensing devices and implements intelligent identification and management. The information sensing devices include infrared sensors, GPS and laser scanner devices. They are all connected to the Internet to implement remote perception and control. IOT is widely applied in intelligent transportation, environment protection, government work, public security, smart home, intelligent fire control, industrial monitoring, elderly care, personal health, etc.

With that in mind, we set our main objective to create a smart IoT system for smart home applications.

## Basic Idea

The system consists of 2 devices: a computer and DAQ USB-6255 device.

The computer will acquire sensor data, process and send signal to control the DAQ device. DAQ device is used to read Passive InfRated sensor (PIR) data, control LED and fan.

On computer, we can control DAQ device in 3 modes:

* Schedule mode: create schedules for fan and LED (FLED)[[1]](#footnote-1) to work automatically at a defined time stamp
* Manual mode: allows users to control FLED manually in 2 ways:
  + On Labview Front panel
  + By finger using Optical Recognition Module (ORM)
* Realtime mode: Read data from PIR sensor and control FLED automatically.

# Section 3: PROJECT DETAILS

## Hardware

### Electronic parts

#### USB-6255

A picture containing text, electronics, projector

Description automatically generatedUSB-6255 is an 80 AI (16-Bit, 1.25 MS/s), 2 AO (2.86 MS/s), 24 DIO USB Multifunction I/O Device—The USB‑6255 offers analog I/O, digital I/O, two 32-bit counters/timers, and analog and digital triggering. The device delivers low-cost, reliable DAQ capabilities in a wide range of applications from simple applications in laboratory automation, research, design verification/test, and manufacturing test.

Figure 1. USB-6255 Device

#### HC-SR501

HC-SR501 is based on infrared technology, automatic control module, using Germany imported LHI778 probe design, high sensitivity, high reliability, ultra-low-voltage operating mode, widely used in various auto-sensing electrical equipment, especially for battery-powered automatic controlled products.

Figure 2.HC-SR501

#### L298N

The L298N is an integrated monolithic circuit in a 15- lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic level sand drive inductive loads such as relays, solenoids, DC and stepping motors

#### Light Emitting Diode

A light-emitting diode (LED) is a semiconductor device that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons.

#### DC Motor

A DC motor is any of a class of rotary electrical motors that converts direct current (DC) electrical energy into mechanical energy. The most common types rely on the forces produced by induced magnetic fields due to flowing current in the coil.

### Wiring diagram

Diagram, schematic

Description automatically generated

Figure 3. Circuit schematic

We created a simple circuit that reads the PIR sensor input in Terminal 1: AI0 of USB-6255. FLED are controlled by Analog output terminal AO0, AO1 respectively. We use LM298 DC Motor Driver Controller to amplify the signal from DAQ to 9 volt and use it to control the Fan.

### House model design

## Software

### Labview

#### User Interface

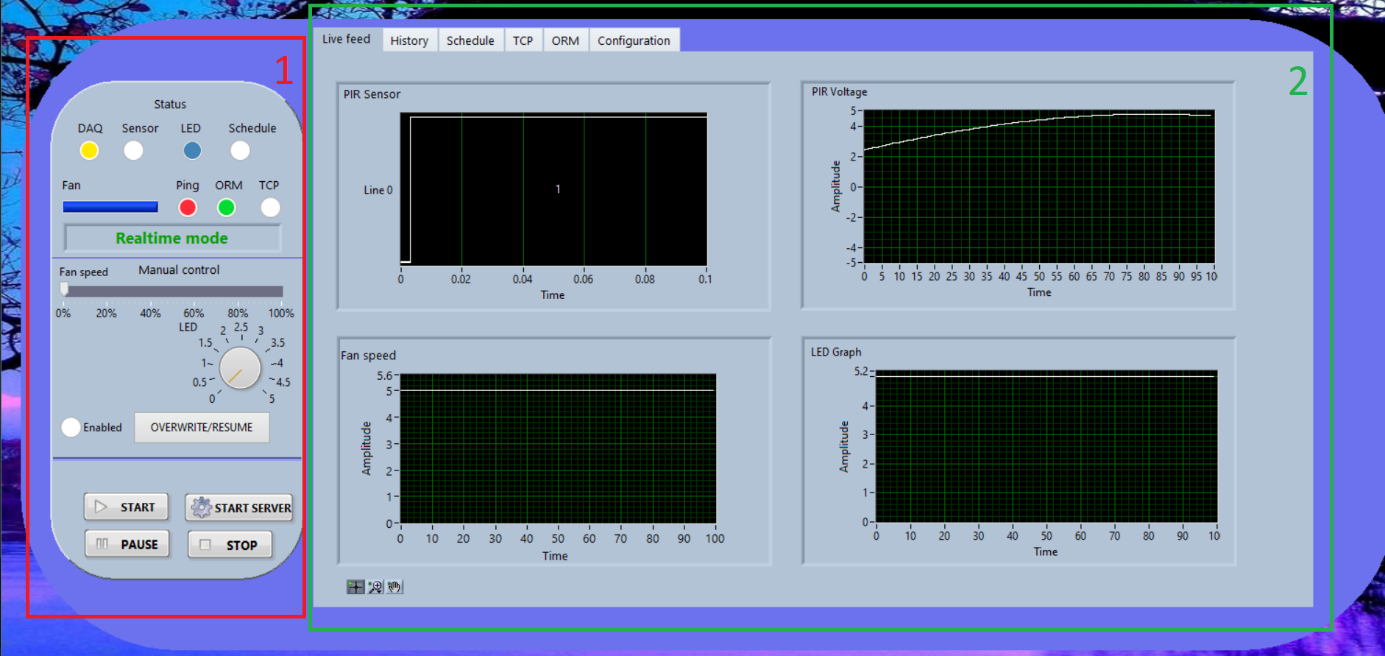


Figure 4. Labview user interface on live feed tab

The user interface (front panel) in UI Main.vi file can control every function of this project. It’s contained 2 parts[[2]](#footnote-2):

* Control panel (1) display status[[3]](#footnote-3) of the program and control FLED manually.
* Tabs (2) contain live feed data from DAQ, TCP server and client information, ORM module and schedule mode control.

##### Manual control

User can control FLED manually by drag and drop the Fan speed slider and LED knob in Control panel. When those value changed, enable light will turn yellow if manual mode not enabled yet. Click on OVERWRITE/RESUME will turn on manual mode and apply change to FLED. Click on it again to resume to the previous mode and end manual mode.

Graphical user interface, text

Description automatically generated with medium confidence

Figure 5. Manual control interface

##### Live feed tab

Live feed tab show latest 100 datapoint in real time:

* PIR voltage is the raw data from DAQ
* PIR Sensor shows processed sensor data, 1 means passed the threshold, 0 mean below.
* Fan speed and LED graph show Output data to FLED.

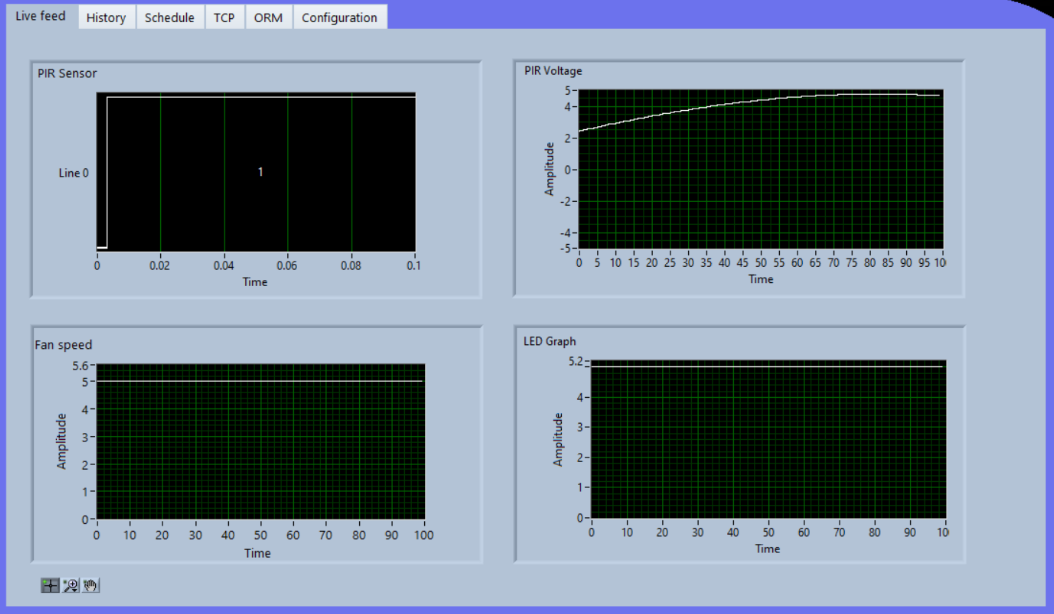


Figure 6. Livefeed tab

##### History tab

History tab show all data from current session, it can also save and load data.

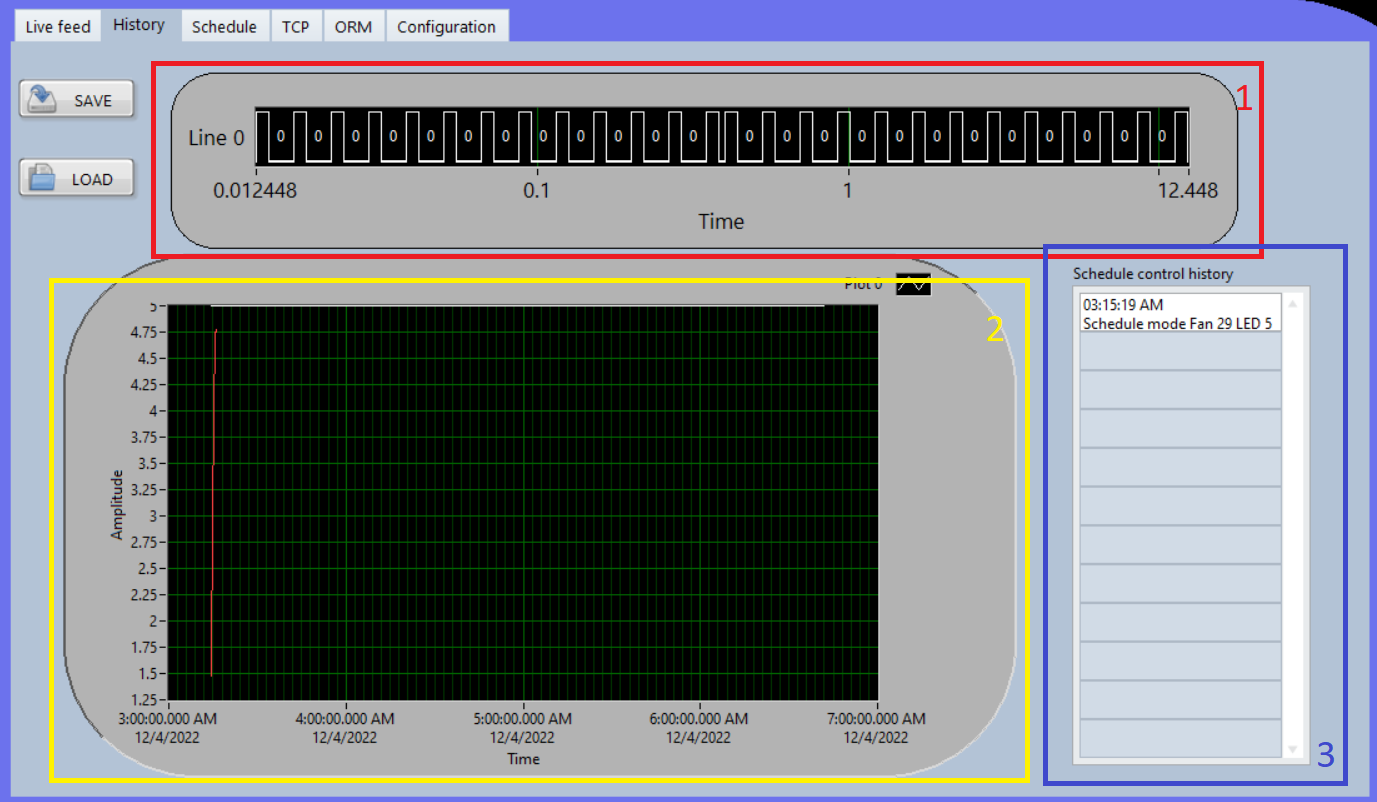


Figure 7. History tab

Graph (1) is PIR sensor data processed, (2) contain 2 lines: Red line is PIR sensor voltage level, white is FLED output data. (3) data table past schedule that executed.

##### Schedule tab

On schedule tab, user can create, modify, delete schedule. Available action are: Light on, Light off, Fan on, Fan off, All on, All off. After created, the schedule will appear on the pending list below in execute time order. Switch on Enable button will change program to schedule mode and each pending schedule will display their time till activation[[4]](#footnote-4). After activated, the schedule will be deleted.

Graphical user interface, application

Description automatically generated

Figure 8. Schedule tab - Disabled

Graphical user interface, application

Description automatically generated

Figure 9. Schedule tab – Activated

##### TCP tab

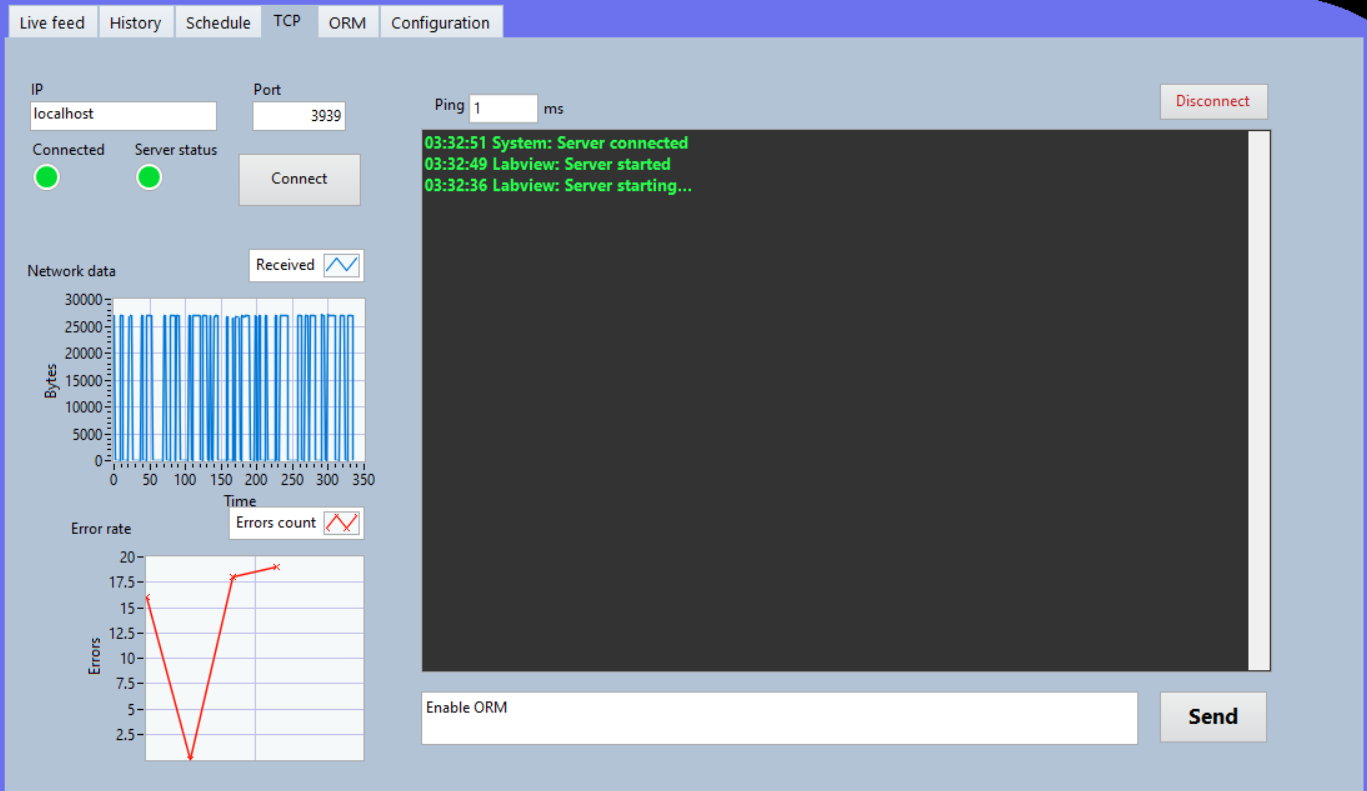


Figure 10. TCP in action

TCP tab shows TCP connection status between Python server and Labview.

##### ORM tab

Graphical user interface, application

Description automatically generated

Figure 11. ORM no signal

ORM tab show MJPEG video stream from python server through ’OMF’ command. Fan value and Light value is from server through ’OMV’ command.

##### Configuration tab

Configuration tabcontain all setting available:

* DAQ device and PIR input terminal choose
* PIR threshold, time before Fan turn off. etc.

Graphical user interface, application

Description automatically generated

Figure 12. Configuration tab – default value

#### Labview block diagram

##### Program structure

This project contain 1 main VI and 21 subVIs, along with 7 typedef. Files and 1 global variable file.

Graphical user interface, text, application, chat or text message

Description automatically generatedA picture containing table

Description automatically generated

Figure 13. Typedef list

Figure 14.Main VI, subVIs list

##### Main VI block diagram

We can break the block diagram into 2 parts for better demonstration: Main block (2) and an event handler (1).Diagram

Description automatically generated

Figure 15. Main UI block diagram

###### Main block

The main program is Flat sequence structure. It contain 3 frame which execute in order from left to right.

Diagram

Description automatically generatedSetup frame

Figure 16. A part of setup frame

The first frame to set up all required variables for each run. This frame will run each time the user presses the Run button.

Variable include: Button, Indicator color, Global ignore error list code, Data value ref, Initial video feed image.

This also set the default mode to Realtime mode.

Simulation mode handling

Diagram

Description automatically generated

Figure 17. Frame 2

The second frame is to check DAQ connection and determine whether to run in simulation mode or not: It tries to send a signal to turn on FLED to the DAQ, the error out from DAQ write wired to simulation check to handle the error.[[5]](#footnote-5)

Diagram

Description automatically generated

Diagram 1. Simulation mode

Program main loop

Graphical user interface, diagram

Description automatically generated

Figure 18. Main while loop

Inside the last frame is 4 case structure, when the user run the program, the first 3 case will execute in parallel with each other:

Case (1) is for compute and process schedule.

Diagram

Description automatically generated

Figure 19. Case (1) - Schedule

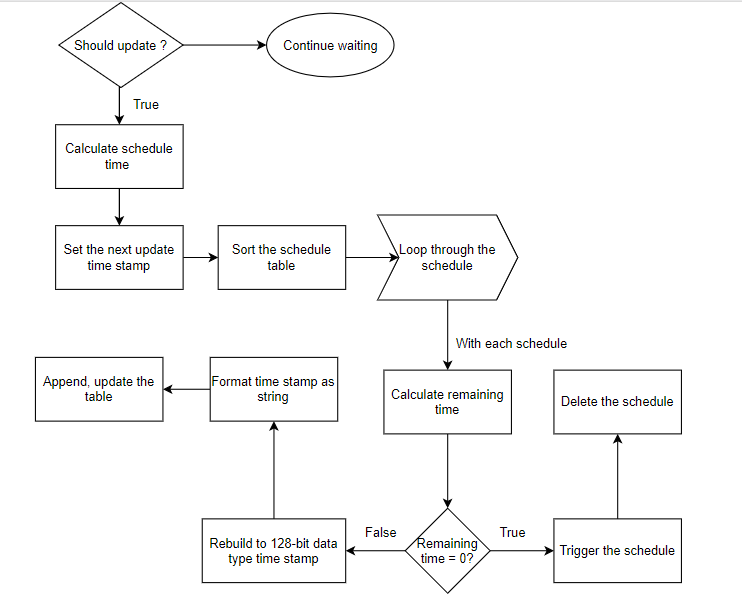


Diagram 2. Update schedule

Case (2) read the sensor data continuously:

* The data from DAQ was then sent to 3 independent sections: Manipulate Output sub VI, show waveform on Live feed tab, save data on history tab.
* Manipulate sub VI handle everything needed to control light and fan.
* Set waveform sub VI create and combine data to show as waveform on Live feed tab.

Diagram, schematic

Description automatically generated

Figure 20. Realtime mode

Diagram

Description automatically generated

Diagram 3. Realtime data acquisition

Case (3) is TCP client data handling block. This case controls all TCP data in and out from the server to Labview client.

A picture containing graphical user interface

Description automatically generated

Figure 21. TCP ready state

Each time we receive data from the server, we need to check for error in connection and data integrity. After verify the data, OK code is send back to the server.

Diagram, schematic

Description automatically generated

Figure 22. TCP data filter, classify

If we discover any error with the connection or data doesn’t match, TCP connection is restarted.

Graphical user interface

Description automatically generated with medium confidence

Figure 23. TCP error handing, Network stats acquire

Ping graph is calculated by subtracting the time when command ends with the command start time. Error rate is the time connection restart per 2000ms.

Diagram

Description automatically generated

Figure 24. TCP client - simplified

TCP restarting procedure:

Diagram

Description automatically generated

Diagram 4. TCP restart

Case (4) is for checking server started state.

###### Event block

The event block contains 20 Event:

1. Save to file:

Write data from history tab to a file

1. Load file:

Read data from file and display to history tab

1. Stop:

Stop the program

1. Reset value:

Using VI server Invoke node to call Default Vals.Reinit All

1. Restart VI:

Call App.restart Invoke node

1. Simulating:

Set the local simulating state to global

1. Start:

Recover from Pause state by set Running to True

1. Pause:

Temporary pause the program by set Running to false

1. Manual mode resume:

Change current mode to Manual mode and enable manual control

1. Connect:

Get the current control IP, Port and attempt to connect to TCP server

1. Start server:

Call python PythonApplication.py in cmd using System Exec.vi

1. Manual mode:

Change the manual in ORM tab

1. Fan speed, LED value change (joint event):

Manually change the FLED output.

Diagram

Description automatically generated

Diagram 5. FLED value change

1. Clear schedule:

Clear the pending schedule

1. Set schedule:

Create a new schedule

1. Enable schedule:

Change program to schedule mode

1. Schedule LED:

Change the Schedule indicator on control panel

1. Reset device:

Reset the DAQ programmatically

1. Remove schedule:

Remove the schedule with ID

1. ID value change:

Update the Change schedule value when change ID control.

#### SubVIs

##### Error collector

Diagram

Description automatically generated

Figure 25. Error collector

Error collector subVI is used to collect error for Error handler subVI.

##### Error handler

Diagram

Description automatically generated

Figure 26. Error handler

Error handler print out error and ignore error in Ignore error global list

##### Simulation check

Graphical user interface, application, Word

Description automatically generated

Figure 27. Simulation check

This subVI check and prompts the user to turn on simulation mode or not.

Simulation mode is used when DAQ device is not available, this require simulated USB-6255 device in NI MAX.

##### Retry TCP

Diagram

Description automatically generated

Figure 28. Retry TCP

This subVI restarts the TCP connection.

##### Win32 decode IMG stream

A picture containing Word

Description automatically generated

Decode JPEG string from TCP stream using CLFN blocks (ole32, windowscodecs and shlwapi calls)

##### String to GUID Cluster

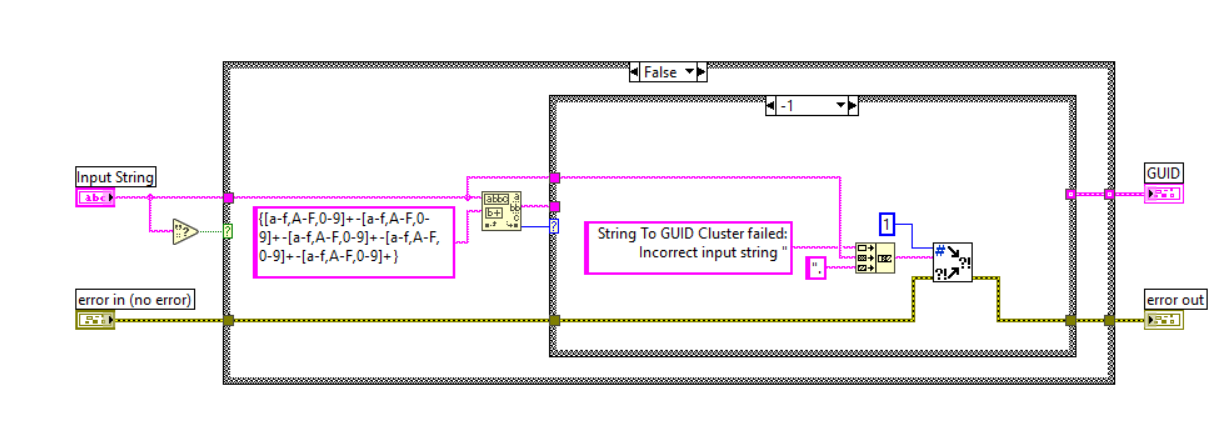


Figure 29. GUID

Part of Win32 decode IMG stream, convert string to GUID cluster.

##### Console Add

Diagram

Description automatically generated

Figure 30. Console

Format the console input and display it in TCP tab, similar to print() function.

##### Get color

Graphical user interface, diagram

Description automatically generated

Figure 31. Get color

Get color cluster from defined color Enum, build into array for easy Property node color change.

##### Change mode

Graphical user interface, application

Description automatically generated

Figure 32. Change mode

Change the current mode to Set mode and Previous mode to current mode.

##### Create schedule

Diagram

Description automatically generated

Figure 33. Create schedule

Create a new schedule , add to global schedules list and sort it.

Diagram

Description automatically generated

Diagram 6

##### Sort schedule

Similar to create schedule subVI, but only loop through the global schedule list and sort it.

##### 

##### 

##### 

##### a

### Python Application

# Section 4: REFERENCES

1. <https://www.ni.com/docs/en-US/bundle/pci-pxi-usb-6255-specs/page/specs.html>
2. <https://www.ni.com/en-vn/support/model.usb-6255.html>
3. <https://forums.ni.com/t5/Example-Code/Programmatically-Reset-VI-Example/ta-p/3532614>
4. <https://forums.ni.com/t5/LabVIEW/Error-363507-LabVIEW-could-not-verify-the-authenticity-of-the/td-p/2881254>

[5]<https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjpm4a1g_76AhUrq1YBHYFkC70QFnoECBEQAQ&url=https%3A%2F%2Fwww.sparkfun.com%2Fdatasheets%2FRobotics%2FL298_H_Bridge.pdf&usg=AOvVaw0wIXqbF0WzC_SUFkv1WstB>

1. Fan and LED [↑](#footnote-ref-1)
2. Figure 4 [↑](#footnote-ref-2)
3. For more information, see Appendix B [↑](#footnote-ref-3)
4. Figure 9 [↑](#footnote-ref-4)
5. For more info see section 2.1.3.3 [↑](#footnote-ref-5)