**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

[Section 1: ABSTRACT 1](#_Toc121087725)

[Section 2: PROJECT OVERVIEW 2](#_Toc121087726)

[1. Inspiration 2](#_Toc121087727)

[2. Basic Idea 2](#_Toc121087728)

[Section 3: PROJECT DETAILS 3](#_Toc121087729)

[1. Hardware 3](#_Toc121087730)

[1.1. Electronic parts 3](#_Toc121087731)

[1.1.1. USB-6255 3](#_Toc121087732)

[1.1.2. HC-SR501 3](#_Toc121087733)

[1.1.3. L298N 3](#_Toc121087734)

[1.1.4. Light Emitting Diode 3](#_Toc121087735)

[1.1.5. DC Motor 3](#_Toc121087736)

[1.2. Wiring diagram 4](#_Toc121087737)

[1.3. House model design 4](#_Toc121087738)

[2. Software 4](#_Toc121087739)

[2.1. Labview 4](#_Toc121087740)

[2.1.1. User Interface 4](#_Toc121087741)

[2.1.1.1. Manual control 5](#_Toc121087742)

[2.1.1.2. Live feed tab 5](#_Toc121087743)

[2.1.1.3. History tab 6](#_Toc121087744)

[2.1.1.4. Schedule tab 7](#_Toc121087745)

[2.1.1.5. TCP tab 8](#_Toc121087746)

[2.1.1.6. ORM tab 8](#_Toc121087747)

[2.1.1.7. Configuration tab 9](#_Toc121087748)

[2.1.2. Labview block diagram 9](#_Toc121087749)

[2.1.2.1. Program structure 9](#_Toc121087750)

[2.1.2.2. Main VI block diagram 10](#_Toc121087751)

[2.1.3. SubVIs 18](#_Toc121087752)

[2.1.3.1. Error collector 19](#_Toc121087753)

[2.1.3.2. Error handler 19](#_Toc121087754)

[2.1.3.3. Simulation check 19](#_Toc121087755)

[2.1.3.4. Retry TCP 20](#_Toc121087756)

[2.1.3.5. Win32 decode IMG stream 20](#_Toc121087757)

[2.1.3.6. String to GUID Cluster 20](#_Toc121087758)

[2.1.3.7. Console Add 21](#_Toc121087759)

[2.1.3.8. Get color 21](#_Toc121087760)

[2.1.3.9. Change mode 21](#_Toc121087761)

[2.1.3.10. Set data waveform 22](#_Toc121087762)

[2.1.3.11. Manipulate output subVI 22](#_Toc121087763)

[2.1.3.12. Create schedule 23](#_Toc121087764)

[2.1.3.13. Sort schedule 24](#_Toc121087765)

[2.1.3.14. Get time Epoch 24](#_Toc121087766)

[2.1.3.15. Convert time & convert time down 25](#_Toc121087767)

[2.2. Python Application 25](#_Toc121087768)

[2.2.1. Purpose 25](#_Toc121087769)

[2.2.2. Environment 25](#_Toc121087770)

[2.2.3. Script structure 25](#_Toc121087771)

[2.2.4. The Optical recognition module 26](#_Toc121087772)

[2.2.5. TCP & Video streaming 26](#_Toc121087773)

[2.2.5.1. Command structure 27](#_Toc121087774)

[2.2.5.2. Video encoding 28](#_Toc121087775)

[2.2.5.3. TCP server handling 28](#_Toc121087776)

[Section 3: RESULT 29](#_Toc121087777)

[Section 4: CONCLUSION 30](#_Toc121087778)

[1. Conclusion 30](#_Toc121087779)

[2. Further development 30](#_Toc121087780)

[Section 5: REFERENCES 31](#_Toc121087781)

[Appendix A. TCP command table 32](#_Toc121087782)

[Appendix B. LabVIEW status indicator 33](#_Toc121087783)

**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 and called it **Smart System utilize passive Infrared sensor to Control Fan and Light in smart home (S.S.I.C.F.L)**, this is an upgrade from mid-term project **using passive Infrared sensor to Control Fan and Light in smart home (I.C.F.L)**.

## 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 InfRared 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 . 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 .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

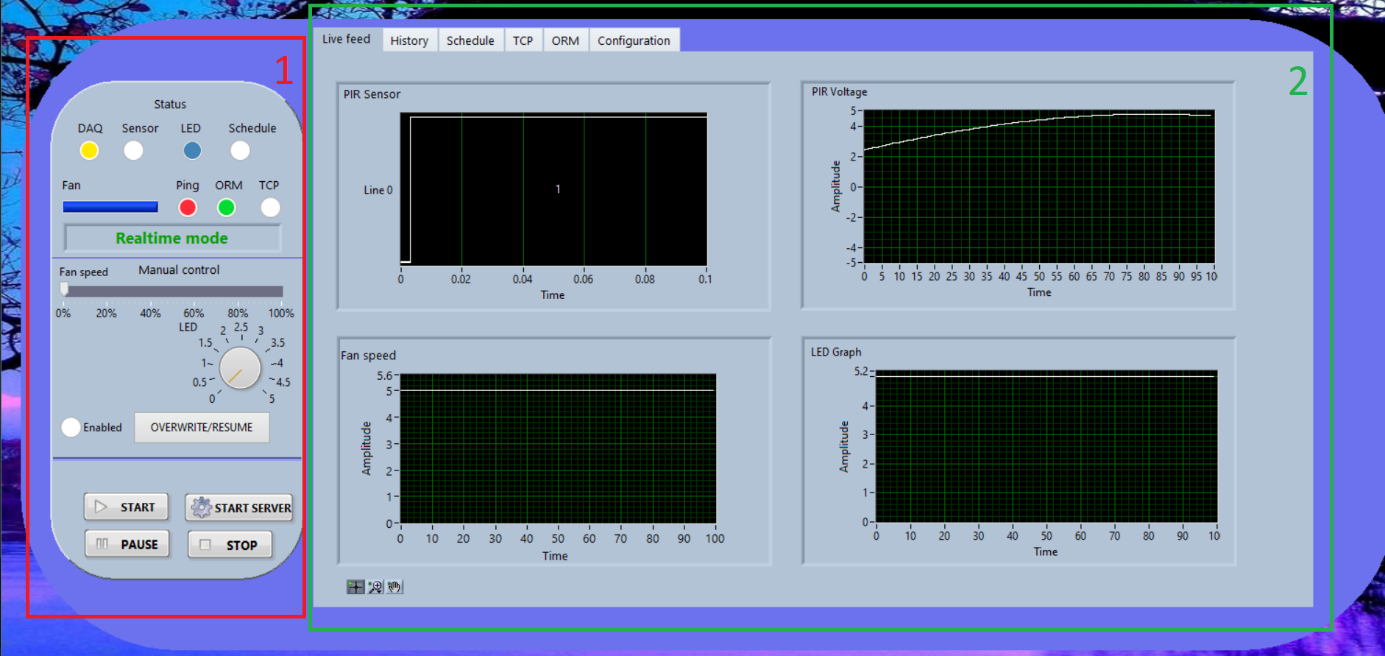


Figure 4. Labview user interface on live feed tab

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.

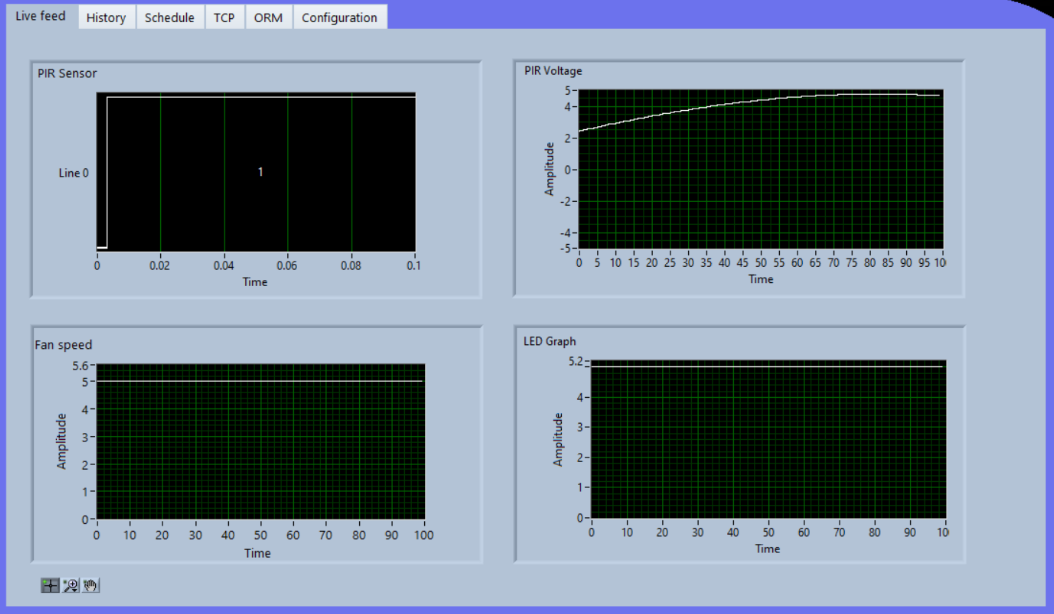


Figure 6. Livefeed tab

##### History tab

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

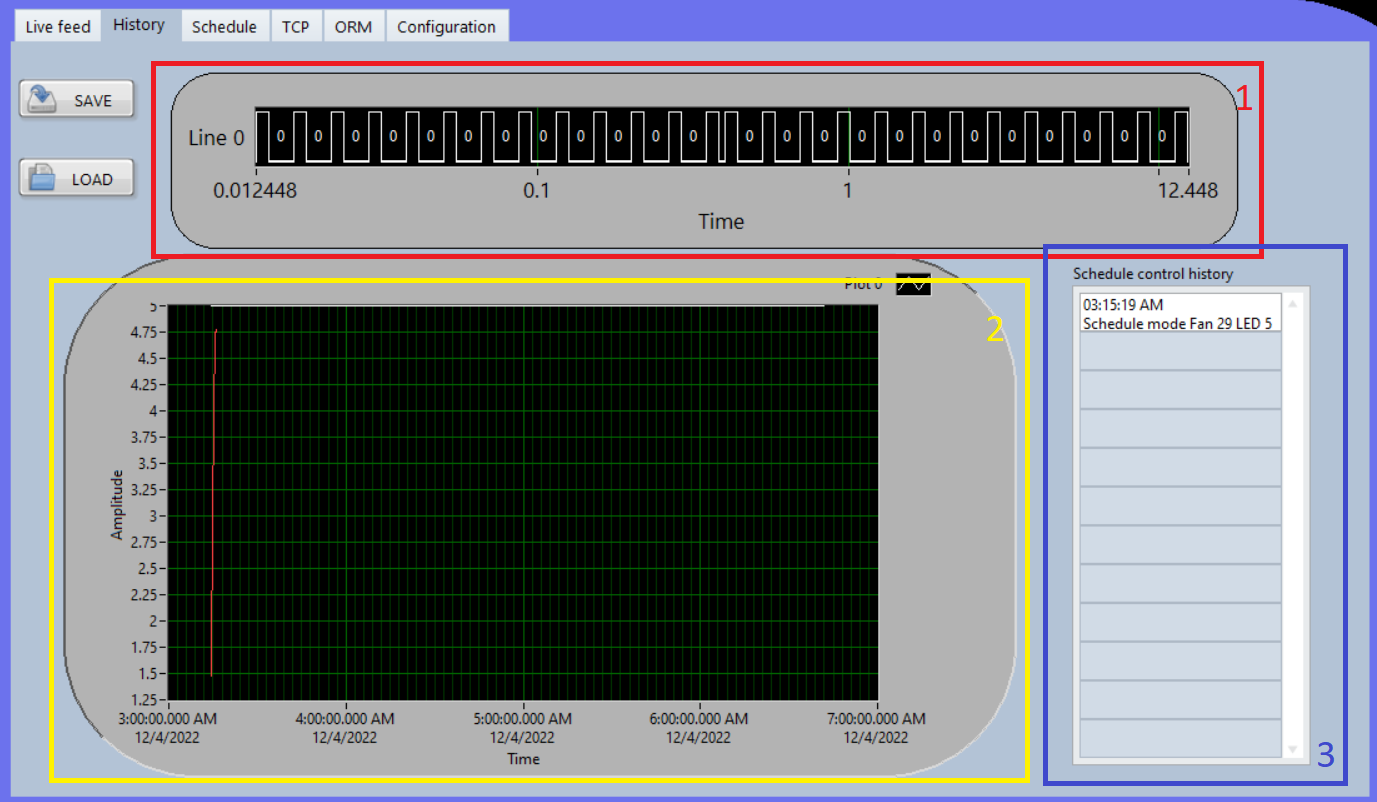


Figure . History tab

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

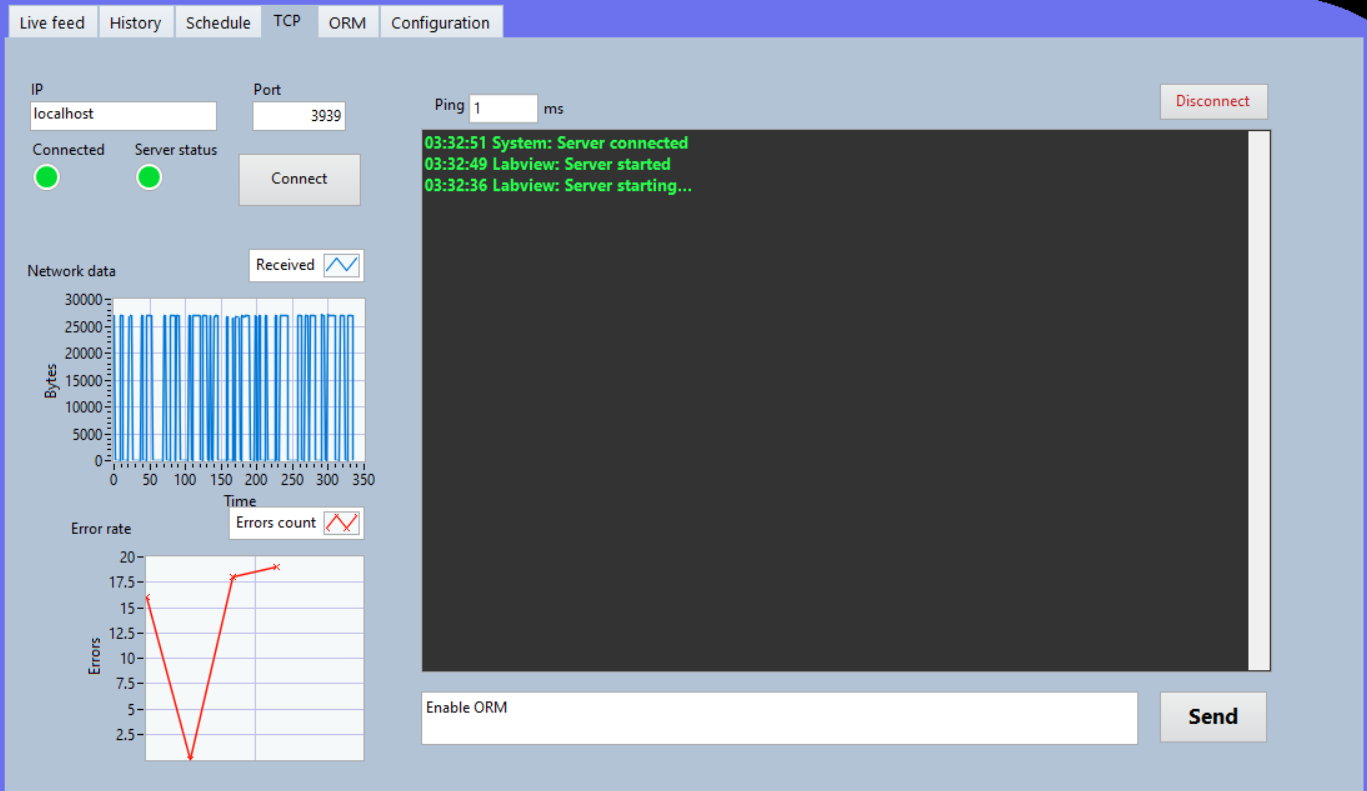


Figure 10. TCP in action

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 . 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 . 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 computing and process schedule.

Diagram

Description automatically generated

Figure 19. Case (1) - Schedule

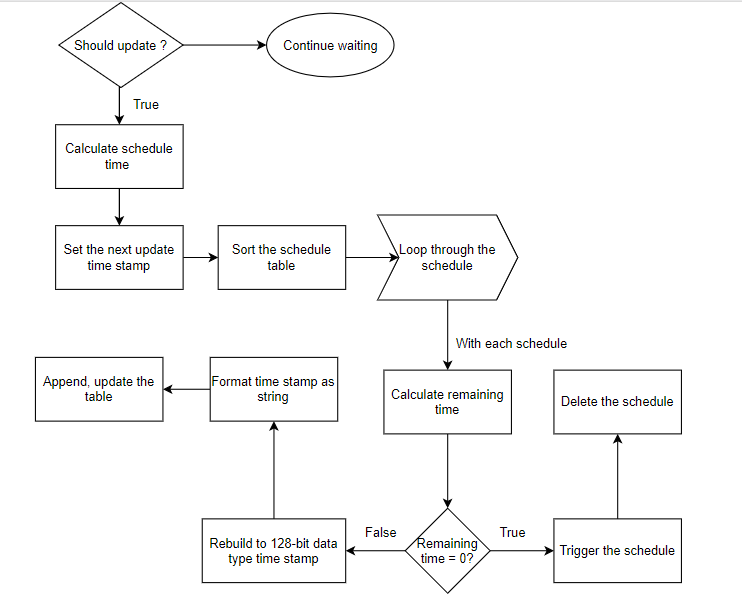


Diagram 2. Update schedule

Case (2) read the sensor data continuously:

* The data from DAQ was then sent to 3 independent sections: Manipulate Output sub VI[[6]](#footnote-6), 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.[[7]](#footnote-7)

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

In our project, SubVIs play an **essential role** which make program up and running perfectly, 60% of Labview our code is in 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

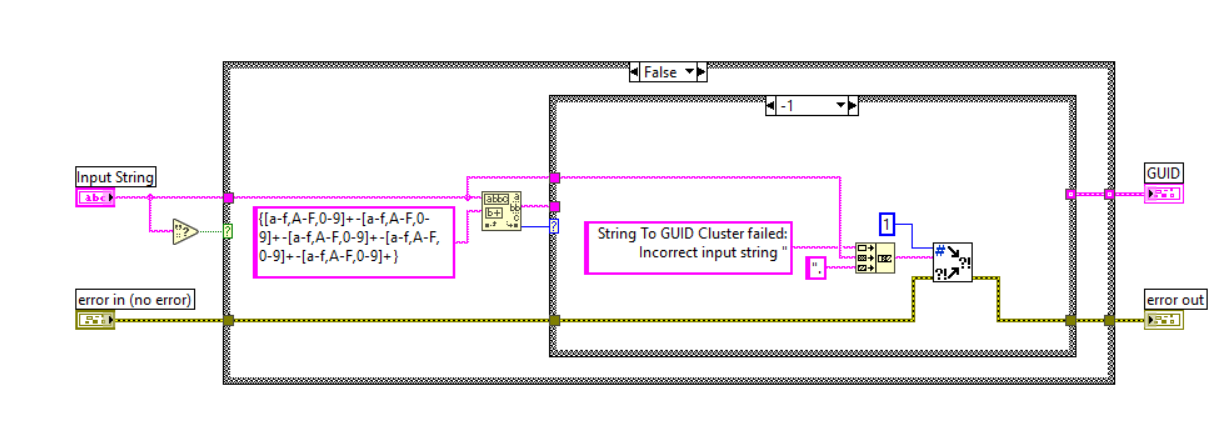


Figure 29. GUID

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.

##### Set data waveform

Diagram

Description automatically generated

Figure . Set data waveform

This subVI get the input waveform, append it with input data and return appended waveform and a sub waveform that contains the first [waveform datapoints] value.

Chart, box and whisker chart

Description automatically generated

##### Manipulate output subVI

This subVI control all write operations to the DAQ, it process data from PIR sensor and calculates time to turn FLED on/off, speed, how long it stay on, voltage to write.

Diagram

Description automatically generated

Diagram 6. Manipulate output subVI - simplified

##### Create schedule

Diagram

Description automatically generated

Figure 34. Create schedule

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

Diagram

Description automatically generated

Diagram 7

##### Sort schedule

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

##### Get time Epoch

Diagram, schematic

Description automatically generated

Figure 35. Epoch

When converting Labview 128-bit time stamp data to LVDataTimeRec type, because LabVIEW timestamp significant 64 bits interpreted as a 64-bit signed two's complement integer represents the number of whole seconds after the Epoch, while LVDataTimeRec is a cluster contain time from year 0. So, to get the correct time stamp we need to convert to Epoch. This subVI return 2 LVDataTimeRec: To Epoch and From Epoch.[[8]](#footnote-8)

##### Convert time & convert time down

Diagram, schematic

Description automatically generated

Figure 36. Convert time

This subVI check and converts the LVDataTimeRec value. For example, 62 seconds will be converted to 1 minute 2 seconds.

LVDataTimeRec will be unpacked and converted into an array of 7 elements, with the last element is 0. Then we loop through all element and with each element: compare the value with a pre-defined constant array containing maximum value allowed on each element (in order: second, minute, hour, day of month, month). And if the element exceeds this value, a while loop will subtract the element and increase the next element value by 1. This continues till the end of list.[[9]](#footnote-9)

### Python Application

#### Purpose

In this project, we used Python as a server to run the optical recognition module and send MJPEG video stream to LabVIEW.

#### Environment

We use Python 3.9 (64-bit) on windows 11 machine with 1280x720 30 fps integrated camera.

CPU: Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz, 2496 MHz, 4 Core(s), 8 Logical Processor(s)

Total Physical Memory: 11.9 GB

TensorFlow is configurated to run on: NVDIA GeForce GTX1650ti Integrated card.

#### Script structure

The python script can be described as follows:

Diagram

Description automatically generated

Diagram . Script diagram

Using threading built-in module, we created 2 threads running in parallel with the hand detection loop to handle the network connection.

#### The Optical recognition module

The optical recognition module refers to detect, track and FLED value converting function in the server and client. On the server, OpenCV is used to capture the video stream from the camera. Then we apply Media Pipe solutions to acquire the hand tracking data and determine if it is in control range or not. If a hand is detected, then we draw its position on to the frame and compute the FLED value. After that, send it to the client video feed.

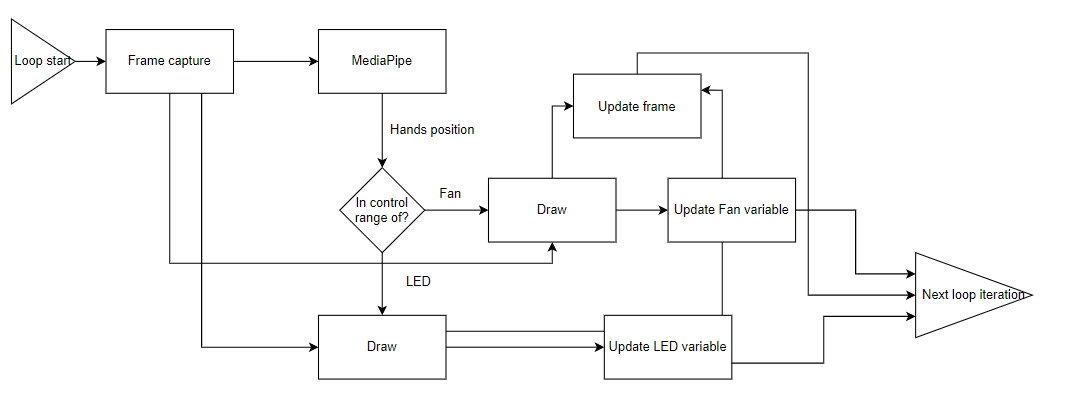


Diagram . ORM module

#### TCP & Video streaming

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol. Therefore, the entire suite is commonly referred to as TCP/IP.

Because LabVIEW doesn’t officially support video streaming from some external source such as MJPEG, we have to build our own command system based on TCP.

##### Command structure

To send the command between Python server and LabVIEW client back and forth, we implemented a 3 stages command:

|  |  |  |
| --- | --- | --- |
| Command data length | Command | Command data |

Each command sent from server will have structure like above. Command data length is 5 bytes of ASCII characters representing the length of Command data in bytes; Command is a 3 bytes of ASCII characters representing the command for the client; Command data is the data that comes with the Command.

Command data length = 5

Example of a command:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0x30 | 0x30 | 0x30 | 0x30 | 0x35 | 0x4f | 0x4d | 0x56 | 0x31 | 0x30 | 0x2c | 0x32 | 0x30 |

We can break the data down:

|  |  |  |  |
| --- | --- | --- | --- |
| Representation | Command data length | Command | Command data |
| Hex | 0x30,0x30,0x30,0x30,0x35 | 0x4f, 0x4d, 0x56 | 0x31, 0x30, 0x2c, 0x32, 0x30 |
| ASCII string | 00005 | OMV | 10,20 |

This command control FLED value of client, set Fan value to 10 and LED value to 20.[[10]](#footnote-10)

*The timed-out event on LabVIEW is set to 50ms, so it’s important to make sure that the command is sent within this time.*

Between each stage, make sure to wait for LabVIEW to respond before sending the next instructions.

##### Video encoding

In order to increase performance, the original frame 1280x720 is downscaled to 360x203 pixels. Then the frame is encoded into JPEG format and stored in server memory. Text

Description automatically generated with medium confidence

Diagram . Video encoding

##### TCP server handling

The TCP server runs in parallel with the ORM. Default, it’s continuously sent 2 commands repeatedly to the client: OMF command and OMV command to update the ORM in client side when connected.

The server also implemented a watchdog timer to retry connection to client when timed out.

# Section 3: RESULT

# Section 4: CONCLUSION

## Conclusion

## Further development

# Section 5: REFERENCES

(n.d.). Retrieved from NI: https://www.ni.com/en-vn/support/documentation/supplemental/08/labview-timestamp-overview.html

*HC-SR501 PIR Motion detector datasheet.* (n.d.). Retrieved from https://www.mpja.com/download/31227sc.pdf

Koon, J. (2019, Feb 13). Retrieved from engineering.com: https://www.engineering.com/story/how-iot-will-change-our-lives

*Motor Driver Module-L298N*. (n.d.). Retrieved from http://wiki.sunfounder.cc/index.php?title=Motor\_Driver\_Module-L298N

NI. (n.d.). Retrieved from Get Components of Time Returned by Get Date/Time in Seconds: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YKVgCAO&l=en-VN

NI. (n.d.). *USB-6255*. Retrieved from https://www.ni.com/en-vn/support/model.usb-6255.html.

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>

# Appendix A. TCP command table

Command to send from server:

|  |  |  |  |
| --- | --- | --- | --- |
| Commands | Hex | Description | Command data |
| LED | 0x4c  0x45  0x44 | Set the LED value | %d |
| FAN | 0x46  0x41  0x4e | Set the fan value | %d |
| OMF | 0x4f  0x4d  0x46 | Send JPEG image stream in memory converted to bytes | %s |
| SCH | 0x53  0x43  0x48 | Create a new schedule  (HH,MM,SS,dd,mm,yy) | %d,%d,%d,%d,%d,%d |
| OMV | 0x4f  0x4d  0x56 | Set the FLED value recognized by optical module (Optical Module Value) | %d,%d |

Client responds:

|  |  |  |
| --- | --- | --- |
| Respond | Hex | Description |
| RDY | 0x52  0x44  0x59 | Client ready to receive command |
| OK0 | 0x4f  0x4b  0x30 | Command data length received, waiting for command |
| OK1 | 0x4f  0x4b  0x31 | Command received, waiting for command data |
| END | 0x45  0x4e  0x44 | Command executed successfully, return to RDY state |
| ERO | 0x45  0x52  0x4f | Data mismatch – Error occurred |

# Appendix B. LabVIEW status indicator

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)
6. See section 2.1.3.11 for more details [↑](#footnote-ref-6)
7. See command structure – section 2.2.5.1 [↑](#footnote-ref-7)
8. https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YKVgCAO&l=en-VN [↑](#footnote-ref-8)
9. The last 6 least important subVI will not be included in this report. For more information, see the source code. [↑](#footnote-ref-9)
10. See appendix A for command table and respond table [↑](#footnote-ref-10)